Volume 107
Number 3
Winter 2021

Journal of the WASHINGTON ACADEMY OF SCIENCES

WASHINGTON ACADEMY OF SCIENCES
1898

12 ISSN 0043-0439 Issued Quarterly at Washington DC

Washington Academy of Sciences

Founded in 1898

Journal of the Washington Academy of Sciences (ISSN 0043-0439)

Published by the Washington Academy of Sciences

email: editor@washacadsci.org

website: www.washacadsci.org

The Journal of the Washington Academy of Sciences

The *Journal* is the official organ of the Academy. It publishes articles on science policy, the history of science, critical reviews, original science research, proceedings of scholarly meetings of its Affiliated Societies, and other items of interest to its members. It is published quarterly. The last issue of the year contains a directory of the current membership of the Academy.

Subscription Rates

Members, fellows, and life members in good standing receive the Journal free of charge. Subscriptions are available on a calendar year basis, payable in advance. Payment must be made in US currency at the following rates.

US and Canada	$30.00
Other Countries	$35.00
Single Copies (when available)	$15.00

Claims for Missing Issues

Claims must be received within 65 days of mailing. Claims will not be allowed if non-delivery was the result of failure to notify the Academy of a change of address.

Notification of Change of Address

Address changes should be sent promptly to the Academy Office. Notification should contain both old and new addresses and zip codes.

Postmaster:

Send address changes to WAS, Rm GL117,
1200 New York Ave. NW
Washington, DC 20005

Academy Office

Washington Academy of Sciences
Room GL117
1200 New York Ave. NW
Washington, DC 20005
Phone: (202) 326-8975

Volume 107
Number 3
Winter 2021

Journal of the WASHINGTON ACADEMY OF SCIENCES

ISSN 0043-0439 Issued Quarterly at Washington DC

EDITOR'S COMMENTS

Presenting the 2021 winter issue of the *Journal of the Washington Academy of Sciences*.

It has been an honor and a privilege to serve as your editor for the Journal for almost twenty years. This winter 2021 issue will be my last as editor. I have learned so much from all the fascinating papers submitted and appreciated all the support. Let me introduce you to our new editor, Ken Baclawski. He is a mathematician at Northeastern University. Please back him as you did me. You may reach him using the same email you used for me: editor@washacadsci.org He will handle the spring 2022 issue and henceforth.

This issue is a special issue dedicated to knowledge graphs. There are five papers by experts discussing the topic. At the end there are the photos of this year's awardees receiving their awards at the annual banquet.

We encourage people to write letters to the editor. Please send by email (editor@washacadsci.org) comments on papers, suggestions for articles, and ideas for what you would like to see in the *Journal*. We also encourage student papers and will help the student learn about writing a scientific paper.

Sethanne Howard

Journal of the Washington Academy of Sciences

Anthropology	Emanuela Appetiti	eappetiti@hotmail.com
Astronomy	Sethanne Howard	sethanneh@msn.com
Behavioral and Social Sciences	Carlos Sluzki	csluzki@gmu.edu
Biology	Poorva Dharkar	poorvadharkar@gmail.com
Chemistry	Deana Jaber	djaber@marymount.edu
Environmental Natural Sciences	Terrell Erickson	terrell.erickson1@wdc.nsda.gov
Health	Robin Stombler	rstombler@auburnstrat.com
History of Medicine	Alain Touwaide	atouwaide@hotmail.com
Operations Research	Michael Katehakis	mnk@rci.rutgers.edu
Science Education	Jim Egenrieder	jim@deepwater.org
Systems Science	Elizabeth Corona	elizabethcorona@gmail.com

Washington Academy of Sciences
1200 New York Avenue
Rm G119
Washington, DC 20005

Please fill in the blanks and send your application to the address above. We will contact you as soon as your application has been reviewed by the Membership Committee. Thank you for your interest in the Washington Academy of Sciences.

(Dr. Mrs. Mr. Ms)

____

Business Address

____

Home Address

____

Email

____

Phone ______________________________________

Cell Phone __________________________________

preferred mailing address   Type of membership

____Business _____Home   ____Regular   ____Student

Schools of Higher Education attended	Degrees	Dates

Present Occupation or Professional Position ____________________
Please list memberships in scientific societies – include office held

____

____

____

Instructions to Authors

1. Deadlines for quarterly submissions are:

Spring – February 1	Fall – August 1
Summer – May 1	Winter – November 1

2. Draft Manuscripts using a word processing program (such as MSWord), not PDF. We do not accept PDF manuscripts.
3. Papers should be 6,000 words or fewer. If there are seven or more graphics, reduce the number of words by 500 for each graphic.
4. Include an abstract of 150-200 words.
5. Use Times New Roman, font size 12.
6. Include two to three sentence bios of the authors.
7. Graphics must be easily resizable by the editor to fit the Journal's page size. Reference the graphic in the text.
8. Use endnotes or footnotes. The bibliography may be in a style considered standard for the discipline or professional field represented by the paper.
9. Submit papers as email attachments to the editor or to editor@washacadsci.org .
10. Include the author's name, affiliation, and contact information – including postal address. Membership in an Academy-affiliated society may also be noted. It is not required.
11. Manuscripts are peer reviewed and become the property of the Washington Academy of Sciences.
12. There are no page charges.

Washington Academy of Sciences Affiliated Institutions

National Institute for Standards & Technology (NIST)
Meadowlark Botanical Gardens
The John W. Kluge Center of the Library of Congress
Potomac Overlook Regional Park
Koshland Science Museum
American Registry of Pathology
Living Oceans Foundation
National Rural Electric Cooperative Association (NRECA)

Modeling with Knowledge Graphs

Kenneth Baclawski*, Gary Berg-Cross*, Cassiopeia Miles***, Douglas Miles***, Todd Schneider^, Ravi Sharma^^

*Northeastern University, **RDA/US Advisory Group, ***Logicmoo Co., ^Engineering Semantics, ^^Senior Enterprise Architect

Abstract

Knowledge graphs represent the confluence of many historical threads that have resulted in a popular new model for managing and retrieving large amounts of data. While a great deal of progress has been made on developing and deploying knowledge graph systems, many challenges remain. This special issue attempts to convey some of the flavor and excitement of this field.

Introduction

KNOWLEDGE GRAPHS (KGs) are lightweight versions of semantic networks that potentially scale to massive data repositories. This special issue expands some of the topics presented in an Ontology Summit on KGs held in 2020 (Baclawski *et al*, 2020; 2020b).

The modern notion of knowledge graph represents the confluence of several ideas and concepts, each of which has its own long history. All the ideas and concepts originate from the development of symbolic reasoning, physically manifested in the creation of symbols in artwork by humans to convey a story. Humans have been depicting non-figurative (i.e., abstract) artwork for at least 40,000 years and possibly much longer (Aubert et al, 2018). We now delve into three of these streams of thought; namely, Mathematics, Ontology and Linguistics.

Mathematics

Thirty thousand years ago, humans kept track of numerical quantities materially by carving slashes on fragments of bone (Cantlon, 2012). The prehistory of the mode of thought called mathematics includes understanding the first thing that may come to mind, which are the symbols we call "numerical terms." This led to the development of numeracy, the earliest forms of which were variables equated to lengths or measurements. Where formerly the concepts were sketched in material substances for tasks such as keeping track of a flock of sheep, numbers were later expressed as symbols in more durable media. As with many cultural developments, their first occurrence was qualitative rather than quantitative (Struik, 2012). As Adam Smith pointed out, numbers are “the most abstract ideas which the human mind is capable of forming.” This development, like many others in mathematics to follow, came only slowly into use.

The evolution of symbolic representations could be understood as exploiting a basic human cognitive ability to make correspondences and use analogies from a concrete experience to a more abstract, modeled domain (Cooke, 2011). By 5,000 BCE there are examples of rudimentary abstraction in the first iconic written numerals (using the cuneiform script) as part of Babylonian culture. Clay tablets from 1800 to 1600 BCE, show that the Babylonians had a knowledge of fractions, algebra, quadratic and cubic equations. There is even evidence of a knowledge of trigonometric functions. (Aaboe, 1991; Robson, 2001)

A thousand years later, there was an abundance of knowledge captured in documents. One of these was the idea of geometry, and the general idea of an abstract proof using logical inferences. Theorems were demonstrably proved by Greek mathematicians starting at around 600 BCE (Boyer, 1968). This work had a major influence on the development of mathematics as a field of study. While the Greeks developed abstractions for geometry and numbers, and depicted geometric notions using graph-like structures, as in Figure 1, the abstract notion of a graph itself is more recent.

The paper written by Leonhard Euler on the Seven Bridges of Königsberg (published in the eighteenth century) is regarded as the first paper in the history of graph theory (Euler, 1736). Since that time, graphs have been used in many domains. One domain that was especially well suited to graphs is chemistry, and it was in the context of chemistry that Sylvester first introduced the word “graph” to mathematics (Sylvester, 1878). Graph theory is now an important branch of mathematics.

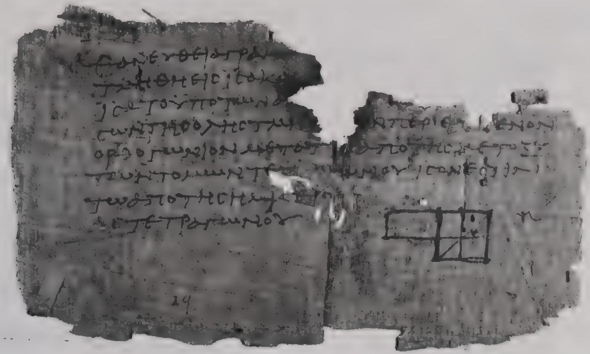


Figure 1: A fragment of Euclid's *Elements*, found at Oxyrhynchus and dated to circa 100 CE

James Sylvester was a colleague of Harvard mathematician Benjamin Peirce, whose son Charles Sanders Peirce developed Existential Graphs (EGs) in 1897. Charles Sanders Peirce and Gottlob Frege independently developed the foundations of what is now known as first-order logic, a notion that has come to dominate modern logic. Peirce's notion of an EG was designed to be fully capable of representing his algebraic notations of first-order and higher-order predicate calculus. This work established that graphs could play a role in the representation of knowledge. The term "knowledge graph" was coined in 1972 (Schneider, 1973).

Ontology

The second stream of ideas is the notion of ontology. In philosophy ontology is the study of concepts such as existence, being, becoming, and reality. The philosophical field of ontology goes back to ancient Indian philosophy in the first millennium BCE (Lochtefeld, 2002; Klostermaier, 2014; Larson, Bhattacharya, and Potter, 2014), and was systematically documented by Aristotle as a major philosophical topic. Aristotle's theories set the standard for logic and ontology with: stable objects as composites of form and matter; ten categories for analyzing, describing, and classifying anything; and logic for specifying patterns and reasoning about them (Sowa, 2016).

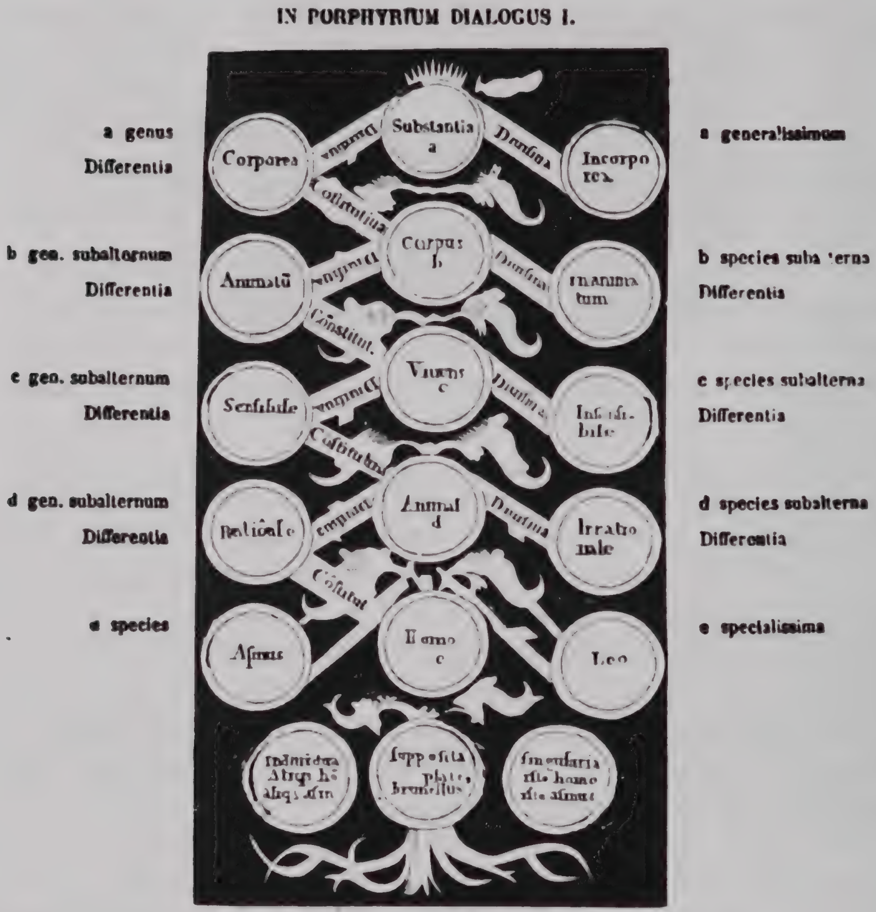


Figure 2: Porphyrian Tree by Boethius

Aristotle's work on ontology led to developments such as the Porphyrian Tree from Porphyry's "Introduction" to Aristotle's categories in the third century CE. Figure 2 shows the Porphyrian Tree as drawn by Boethius from his translation of Porphyry's treatise in the sixth century. Ramon Llull further developed the Porphyrian Tree into a forest of sixteen trees in the thirteenth century, which set the stage for medieval philosophical-theological developments of logic and the problem of universals. The mathematical and philosophical discussions served as an inspiration to the pioneer in symbolic logic, Gottfried Wilhelm Leibniz, and the concepts in the Porphyrian Tree continue to benefit the classification of living organisms.

However, the use of ontology in modern information processing systems only emerged in the mid-1970s. It was at this time that AI researchers began to recognize that knowledge engineering was necessary for building large and powerful AI systems. By the 1980s, the AI community began to use the term "ontology" for a theory of a modeled world as a component of knowledge-based systems. Such ontologies were (and still are) usually based on classes and relationships between them, such as type hierarchies, part-whole relationships, and many others. Although classes and relationships were visualized using graph-like diagrams to help people to understand the ontology, the knowledge-based systems of the time did not describe themselves as being graphs, and the knowledge that was encoded was not regarded as being a "knowledge graph" per se.

Linguistics and Natural Language Processing

A third stream of thought came from the field of linguistics. The interplay of how humans communicate and naturally represent entities and relationships in the world can provide inputs to a knowledge graph, whether the inputs arise from text, images, sound, video, or any other medium. The small sizes and relative simplicity of the earliest data processing activities could be managed using fixed structures and traditional database technology because the linguistic needs were relatively simple. The advent of tools for entity extraction and Natural Language Processing (NLP) that are now easy to obtain and to deploy in applications has resulted in much larger amounts of information with a more complex interplay among the natural language terms. A more flexible data model is needed to deal with this complexity, which is one possible reason why KGs have become so popular.

While much of the work on NLP has been focused on English, support for other languages is now emerging. Recent efforts at generating KGs and answering queries have been made in Sanskrit (Terdalkar and Bhattacharya, 2019), and there are also efforts in other languages, some of which have been very advanced for some time now (Wang, 2013; Wu, 2018).

Confluence

The confluence of the three streams of ideas discussed above was not a single event. The work of Ramon Llull could be regarded as an early example of a notion of knowledge graph that combines aspects of mathematics and ontology. Some other examples are Peirce's work on triadic logic from the 1860s, which influenced the upper-level category system of the KBPedia Knowledge Ontology (KKO) (KBPedia, 2021), and John Sowa's conceptual graph notion that was inspired by semantic networks in Artificial Intelligence (Sowa, 1976).

One of the most significant developments in the history of knowledge graphs was the development of

the Resource Description Framework (RDF), which was first published in 1997 (Guha and Bray, 1997). Although RDF is a graph-based data model, it was not originally designed for KGs; indeed, RDF was not originally intended for data at all, but rather for metadata annotations (RDF, 2014). There are many query languages for RDF, but one of the most commonly used is SPARQL, whose standard was first released in 2008 (SPARQL, 2013).

The publication of Google's Knowledge Graph in 2012 was a major event in the history of KGs (Singhal, 2012). While Google didn't invent KGs, it was the main popularizer, and there has been steady growth of interest in KG research and development in recent years. The papers in this special issue provide an overview of KG system research and development, as well as some of the many challenges of this field.

Introducing the Papers

In this special issue, the first article discusses the issue of why people and organizations should devote resources to capturing and organizing information at all. Matthew West has had a distinguished career in information management and applied ontology. He is the Technical Lead for the UK Digital Twin programme and was awarded an OBE for services to information management in the 2021 New Years Honours List (UK). Matthew West's article proposes that information matters because it is used to support organizational decisions and, when critical information is sharable and structured, it is the key to enabling automation, thereby improving productivity. Knowledge graph systems, in particular, can contribute to business and information processes.

Having discussed why knowledge graphs should be developed and maintained, we then consider how to develop them. In practice, knowledge graphs, especially very large ones, are not constructed all at once but rather incrementally by an iterative process, adding both new data and new metadata, including deeper semantics. In "Issues in incrementally adding better semantics to Knowledge Graphs," Gary Berg-Cross discusses the advantages and issues that arise when knowledge graphs are developed incrementally. Gary Berg-Cross is a Cognitive Psychologist who has published extensively in both Cognitive Psychology and Artificial Intelligence. Most recently he has been involved in various aspects of the Semantic Web, including participating in the Spatial Ontology Community of Practice to provide better semantics to support data sharing as well as vertical and horizontal integration, with special emphasis on the Earth Sciences.

Knowledge graphs are a popular technique for data management in spite of a lack of agreement about what knowledge graphs are. One of the accomplishments of the Ontology Summit 2020 was to reach a community consensus by major contributors to the field of knowledge graphs on a precise mathematical definition of a knowledge graph. In "A Knowledge Graph Data Model and Query Language," Kenneth Baclawski has proposed that the definition of a knowledge graph can be the basis for a new model for data management and retrieval, called the *knowledge graph model*, and he also presents an example of a data language, KGSQL, for this new model. The knowledge graph model has a number of advantages compared with existing graph data models such as RDF and property graphs. KGSQL is syntactically similar to SPARQL, but allows one to take advantage of the new capabilities of the knowledge graph model. Kenneth Baclawski is an Emeritus Associate Professor of Computer Science at Northeastern University.

The special issue ends with a review of the major challenges facing the field of knowledge graph systems in the article "Challenges in the Design, Implementation, Operation and Maintenance of Knowledge Graphs" by Gary Berg-Cross. This article reviews both internal challenges within the KG system lifecycle and external challenges on KG systems. The internal challenges are for designing, populating, refining, deconflicting and maintaining a KG. The external challenges include capturing context, completeness issues, and domain-specific knowledge.

Acknowledgements

We would like to thank Janet Singer, Ram D. Sriram, and Andrea Westerinen for their invaluable help, without which this special issue would not have been possible.

References

Aaboe, A. (1991). "The culture of Babylonia: Babylonian mathematics, astrology, and astronomy". In Boardman, John; Edwards, I. E. S.; Hammond, N. G. L.; Sollberger, E.; Walker, C. B. F. (eds.). *The Assyrian and Babylonian Empires and other States of the Near East, from the Eighth to the Sixth Centuries B.C.* Cambridge University Press. ISBN 0-521-22717-8.

Aubert, M. *et al*. (2018). "Pleistocene cave art from Sulawesi, Indonesia," *Nature* volume 514, pages 223–227.

Baclawski, K., Bennett, M., Berg-Cross, G., Fritzsche, D., Sharma, R., Singer, J., Sowa, J., Sriram, R.D., Whitten, D. (2020). Ontology Summit 2020 Communiqué: Knowledge Graphs. In *Applied Ontology*, 16(2), 229–247.

Baclawski, K., Sowa, J., Sharma, R., Berg-Cross, G., Sriram, R., Singer, J. (2020b). Introduction and Overview. In Ontology Summit 2020: Knowledge Graphs. Retrieved 22 October 2021 from https://go.aws/38Sv50z.

Boyer, C. (1968). *A history of mathematics*. John Wiley & Sons, Inc., New York.

Cantlon, Jessica F. (2012). "Math, monkeys, and the developing brain." In Proceedings *of the National Academy of Sciences* 109.Supplement 1: 10725-10732.

Cooke, Roger L. (2011). The history of mathematics: A brief course. John Wiley & Sons.

Euler, L. (1736). Solutio problematis ad geometriam situs pertinentis. In *Comment. Acad. Sci. U. Petrop* 8, pp. 128-140.

Guha, R.V. and Bray, T. (1997). “Meta Content Framework using XML,” June 6, 1997. Retrieved 22 October 2021 from https://bit.ly/3b4pxTy.

KBpedia (2021). Open Standards, KBpedia. Retrieved 22 October 2021 from http://kbpedia.com/standards/

Klostermaier, K.K. (2014). A Concise Encyclopedia of Hinduism. Oneworld Publications, Oxford. p. 112. ISBN 978-1-78074-672-2.

Larson, G. J., R. S. Bhattacharya, and K. Potter, eds. (2014). "Samkhya." Pp. 3-11 in The Encyclopedia of Indian Philosophies 4. Princeton University Press. ISBN 978-0-691-60441-1.

Lochtefeld, James (2002), “Nirukta” in The Illustrated Encyclopedia of Hinduism, Vol. 2: N-Z, Rosen Publishing, ISBN 0-8239-2287-1, page 476.

Robson, E. (2001) “Neither Sherlock Holmes nor Babylon: a reassessment of Plimpton 322,” *Historia Math.* 28 (3), p. 202.

RDF (2014). RDF 1.1 Concepts and Abstract Syntax. Retrieved 22 October 2021 from www.w3.org/TR/rdf11-concepts/

Schneider, E.W. (1973). Course Modularization Applied: The Interface System and Its Implications For Sequence Control and Data Analysis. In *Association for the Development of Instructional Systems (ADIS)*, Chicago, Illinois, April 1972.

Singhal, Amit (2012). Introducing the Knowledge Graph: things, not strings. *Official Google Blog.* Retrieved 22 October 2021 from https://bit.ly/3Cjjsi0.

Sowa, J. (1976). “Conceptual Graphs for a Data Base Interface.” In *IBM* Journal of Research and Development 20(4): 336-357.

Sowa. J. (2016) Aristotle’s Patterns Of Logic and Ontology. Retrieved 22 October 2021 from https://bit.ly/2XHn4vu.

SPARQL (2013). SPARQL 1.1 Query Language. Retrieved 22 October 2021 from https://bit.ly/3ppnB0t.

Struik, Dirk J. (2012). A concise history of mathematics. Courier Corporation.

Sylvester, J.J. (1878) Chemistry and Algebra, *Nature*. 17 (432): 284.

Terdalkar, H. and Bhattacharya, A. (2019). Framework for Question-Answering in Sanskrit through Automated Construction of Knowledge Graphs. Retrieved 22 October 2021 from https://bit.ly/3m2clFc.

Wang, Z., Li, J., Wang, Z., Li, S., Li, M., Zhang, D., Si, Y., Liu, Y., Zhang, P., Tang, J. (2013). XLore: A Large-scale English-Chinese Bilingual Knowledge Graph. Retrieved 22 October 2021 from https://bit.ly/3Gbemab.

Wu, T., Qi, G., Li, C., Wang, M. (2018). A Survey of Techniques for Constructing Chinese Knowledge Graphs and Their Applications. In *Sustainability* 10(9). ISSN 2071-1050. Retrieved 22 October from https://bit.ly/2ZdjVDU.

Information Management Framework: Quality Basics

Matthew West, OBE

University of Cambridge

Abstract

Information and knowledge are important for organizations. This article discusses why information is important and how to ensure that information is fit for its purposes. It is proposed that information matters because it is used to support organizational decisions. Consequently, poor quality information can increase the risk of mistakes and disasters, and reducing mistakes directly improves productivity. The article also discusses how holding critical information as sharable, structured data is a key enabler of automation, which is another way of improving productivity. We provide ways of looking at information which are useful in assessing and understanding how to manage it, including the role of information services. Knowledge graph systems are a useful way to hold information as data so it can contribute to business and information processes. We outline the basic elements of information management including (1) Identifying information requirements, (2) The critical properties of information, (3) The information lifecycle, and (4) Applying quality management to information.

Introduction

Knowledge graph systems are a popular technology for storage and retrieval of information as data by businesses. In this article we examine the reasons for the use of knowledge graph systems and for information in businesses in general. The primary use of information in business is to support decisions so that outcomes are better, and the risk of mistakes is reduced. Despite this it is not difficult to find examples where poor quality information has led to mistakes having been made that has led to embarrassment at best and disasters at worst. The cause usually found for mistakes and disasters is that the decision was based on a lack of information, or on poor quality information, indicating that information is managed poorly, or not at all. Reducing mistakes directly improves productivity. Another important driver in improving productivity is automation – getting computers to do what people do at present. Computers are good at doing dull repetitive tasks that can be pre-determined; and coincidentally these tend to be tasks people are not so good at, which can lead to mistakes. A key requisite for automation is for information to be held as data, such as in knowledge graphs, so it can be conveniently processed. So a key part of achieving automation is not just moving data into an electronic form for people to access, but turning it into data so that computers can process it.

One of the problems that software and data can have in a business setting is the difficulty of properly accounting for their costs and benefits. For some companies, software and data are products that are sold to other companies and to individuals, in which case software and data have direct revenues. However, when a company is using software and data within its own business processes and products, it can be difficult to account for the benefits of software and data. From an accounting point of view the value of software and data is an intangible asset. An intangible asset is an asset that lacks physical or financial substance. Examples are software, patents, copyright, and trademarks, as well as data. A reduction in cost to a business, such as improving productivity or preventing disasters, will not appear as revenue on a balance sheet. As a result, software and data might only appear as costs with no corresponding benefits. Nevertheless, the costs of acquiring and managing software and data are eligible to be recognized as intangible assets provided they are (1) identifiable, (2) controlled by the organization, and (3) have future economic benefit such as revenues or decreased costs. The future economic benefits of intangible assets can be difficult to quantify because they create value in combination with other production factors. (Lev and Daum, 2004) Accordingly, it is important to specify the roles and purposes of software and data as contributors to other business processes.

Here are some simple criteria in the form of questions that will help one to determine whether information is being well managed by an organization:

1. Is there a clear understanding at the top of the organization that information needs to be managed so it supports decisions, and is there commitment to providing the resources to do so? This is the level at which it is most appropriate for an organization to identify the contribution of information to business processes so that the cost of information is properly accounted.
2. Is there a process model that identifies the decisions taken in the organization together with documentation of the information required to support them and its quality requirements? Specifying and

documenting process models are best done using structured information models.

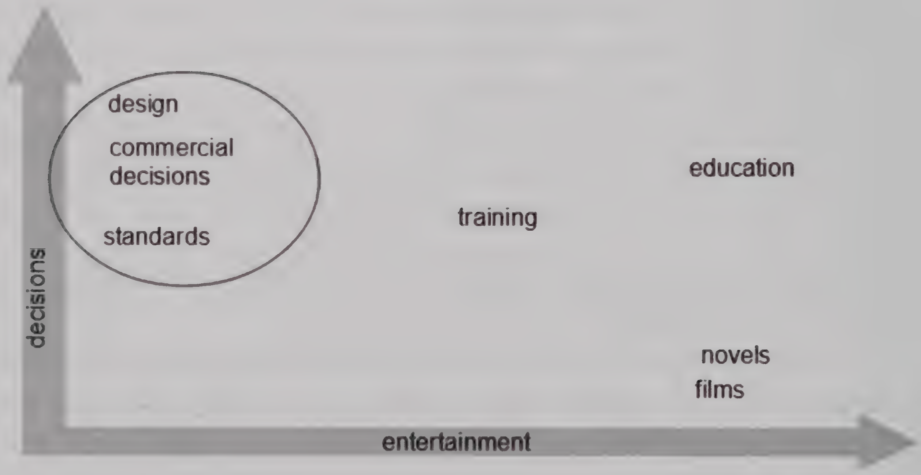
3. Is there a drive to automate processes with information as structured data as a key enabler of the automation? While traditional spreadsheets and database systems can be used for process modeling, the flexibility of knowledge graphs make them well suited for the purpose of specifying process models and documenting decisions.
4. Do those who create information know the uses to which it will be put and the quality requirements of those uses? This is not only important for maintaining the quality of information but also for allocating sufficient resources and manpower.
5. Do those involved in the creation, use, and management of information have tasks and targets set for their performance of those roles? Having formally specified processes can contribute to this criterion.
6. Does shared data conform to a common data model and reference data so that data from different sources can be brought together and used without first wrangling it into a usable form? This is especially difficult for large organizations or organizations that were formed by merging other organizations.
7. Is the performance of the information management process measured so that problems with information can be identified and fixed? If performance measurements are informally specified and evaluated by people, problems can be missed or discovered too late. Automated process measurement can be more efficient and faster than manual techniques.
8. Is there an improvement process in place for the information management process so that when defects or near misses occur, the root causes are identified and corrected? This is important for all management processes.

The purpose of this article is to set out the basics of information quality management: the management of knowledge so that it is fit-for-purpose for the decisions it supports whether automated or manual, especially for knowledge graph systems. This document is aimed at all those who create, use and manage information to support decisions, so they can understand their part in ensuring decisions are supported by information that is fit-for-purpose.

Why Bother with Information?

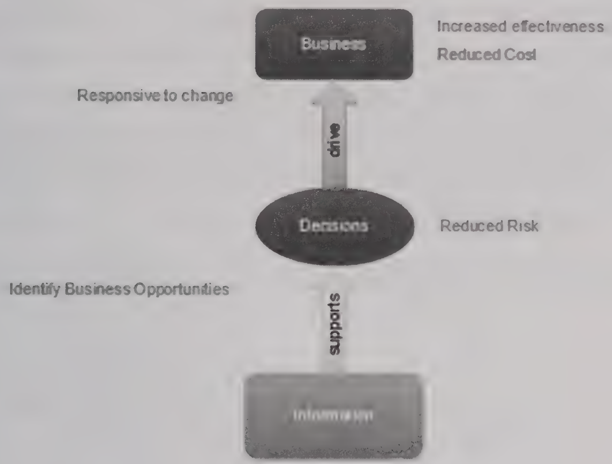
Obtaining and managing information can be expensive, especially if care is taken to ensure that the information is of high quality. Assuming that one is using information directly, and not just selling it as a product, one can classify the uses of information along two axes as in Figure 1. One axis is the degree to which the information is being used for education and entertainment, and the other axis is the degree to which the information is being used to support decisions. When information is being used for entertainment and not for decisions (*i.e.*, the lower right corner of the Figure 1), then information can be regarded as a product. When this is the case, traditional product management and accounting practices are appropriate. For other uses of information shown in Figure 1, information will be used for making decisions, at least to some degree, and it is this use of information that is the focus of this document.

Figure 1 Matrix of information uses



Information is used to support decisions which drives a business. Most obviously it helps reduce the risk of making a mistake – which reduces costs of fixing mistakes and improves effectiveness, but in doing so this also helps to identify business opportunities, and makes your organization more responsive to change, see Figure 2. When we are talking about mistakes, it

Figure 2: Why bother with information?

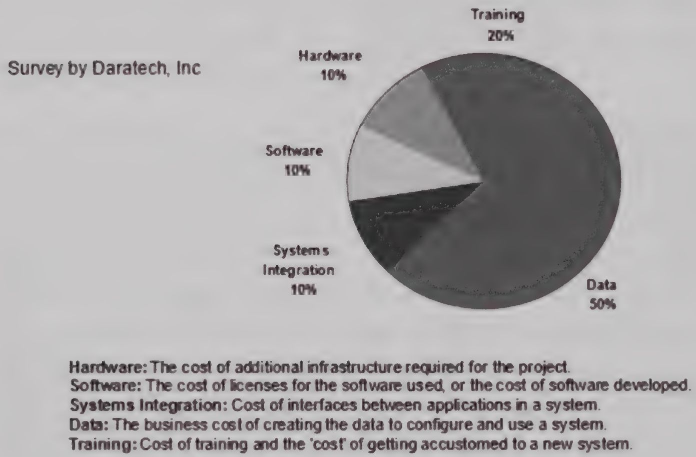


is important to understand the range this covers. A mistake might be something as small as ordering too many paper clips, where the waste from the mistake is insignificant, or it might be one of a series of mistakes that together lead to a disaster like Grenfell Tower where seventy-two lives were lost and the cost is already in the billions of dollars/euros/pounds. Eliminating waste and mistakes leads directly to improvements in productivity. For the Grenfell Tower disaster, it has already been acknowledged that a key element was the lack of a "Golden Thread" of information. (Hackitt, 2018) However, mistakes in general are not deliberate, and usually the result of the information that would have helped avoid the mistake being either unavailable or inaccurate, or being discovered too late to prevent the mistake from occurring.

As discussed in the introduction, properly accounting for the benefits of allocating resources to improving the quality and timeliness of information can be difficult. However, it is not entirely a matter of guesswork, as there is a substantial literature on risk monitoring, mitigation and management for software engineering (Pressman, 2001), which can also be applied to data engineering.

Another issue that is not widely understood is the cost of data, relative to other elements of an information system. This is largely because the cost of data is usually unaccounted for. However, in the 1990s Daratech Inc. did a survey of the costs of different elements of developing engineering design information using Computer Aided Design. Figure 3 shows the breakdown. Note that half the cost was the creation of the data itself. This comes as a surprise to many, but it should not be really, since the principal product of the design process is the design, which is information. It is just not accounted for as such. There are some unfortunate consequences of this lack of recognition, which include that much effort will be put into reducing the costs of the other elements, even when this will result in increases in costs in creating the design data itself.

Figure 3: The most expensive component of an information system is the data.



Hardware: The cost of additional infrastructure required for the project.
Software: The cost of licenses for the software used, or the cost of software developed.
Systems Integration: Cost of interfaces between applications in a system.
Data: The business cost of creating the data to configure and use a system.
Training: Cost of training and the 'cost' of getting accustomed to a new system.

Some Ways of Looking at Information

In this section we look at some different ways of looking at and classifying information that are largely orthogonal, with each being potentially useful in particular situations.

Facts, information, knowledge, wisdom, data

In Figure 4 we show a popular distinction that is made between levels of information: facts, information, knowledge, and wisdom. Rather than give formal definitions of what these things are, we describe the usage we see made of these terms in practice.

Information

We see the term information being used with two senses:

1. As a catch-all for all the different ways of talking about, well, information including those presented here.
2. In a narrower sense as the information that informs a decision (the sense in Figure 4).

The context usually makes which sense is intended clear. Information (in the broad sense) can be structured or unstructured. Structured information is usually referred to as data (see below). Unstructured information takes the form of pictures, documents and drawings that are intended for human consumption rather than computer processing.

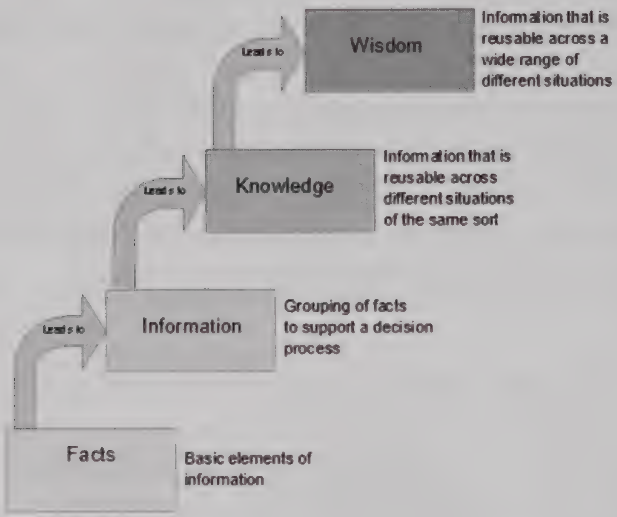
Data

Once upon a time, data (or perhaps more properly datum) was a synonym for fact. However, in recent decades it has come to mean information held in structured form, such as a database or knowledge graph, where some of the meaning of the data is determined by the structure in the form of a data model or ontology. We use that sense here. Also, we use data as both singular and plural, as most people do unless they think really hard about it. Data is relatively easy for computers to process. Computers are good at doing dull repetitive tasks that can be pre-determined, and coincidentally these tend to be tasks people are not so good at. Thus a key part of achieving automation is having information as data, that is moving beyond just holding information in an electronic form for people to access, but turning it into data in databases or knowledge graphs so that computers can process it. This is a key step in digital transformation.

Facts

A fact is the smallest piece of information, such as a triple in a knowledge graph or a sentence in a document. In many charts that show these levels, this is called data, but for us data has another more useful usage, and fact is a good alternative name.

Figure 4: Levels of information.


Wisdom
Information that is reusable across a wide range of different situations
Leads to
Knowledge
Information that is reusable across different situations of the same sort
Leads to
Information
Grouping of facts to support a decision process
Leads to
Facts
Basic elements of information

Knowledge

Knowledge is information that is reusable across multiple situations of the same sort. Publicly available information on the Web is an example of knowledge. Within a business, master and reference data are examples of knowledge, as are Key Performance Indicator values and designs. While any data representation technique can be used to store and access knowledge, knowledge graph systems are well suited for information that is reusable across multiple situations. (Baclawski, 2021)

Wisdom

Wisdom is knowledge of the widest applicability. It is usually about people and understanding how they will behave. Wisdom is very long lasting and not usually found in databases.

It is useful to know which level you are talking about, and when we come to consider the information lifecycle, we will look at how these interact with each other.

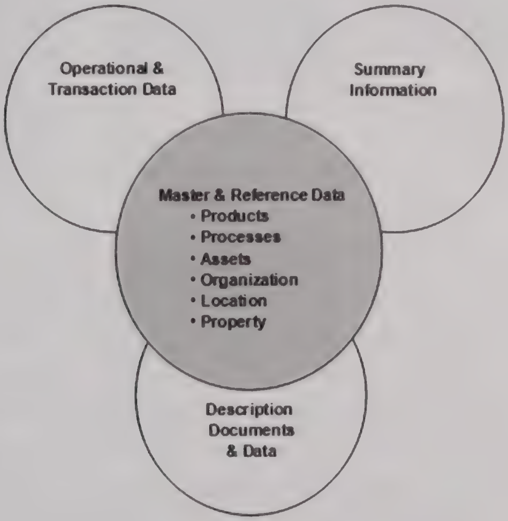
The Mickey Mouse Diagram

Another way to look at information, but in particular data, is illustrated in Figure 5, usually known as the Mickey Mouse diagram. In this view of data we have:

- Operational and transaction data, including measurements and raw transactions such as sales transactions.
- Summary information, used for management purposes and that is sliced and diced for analysis of performance.
- Master and reference data, that is used to slice and dice the summary information, and is about such things as products, processes, assets, organizations, locations and properties.
- Description documents and data, that describe what the master and reference data represent.

Generally, these different types of information have different characteristics and are managed differently and separately, held together by the master and reference data.

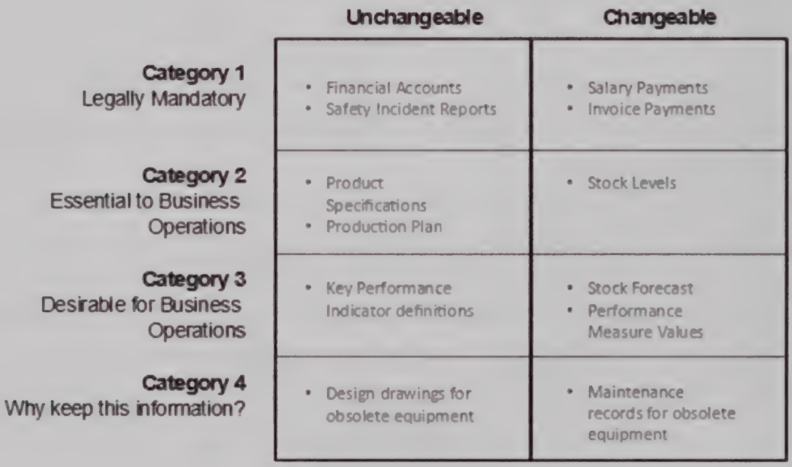
Figure 5: Different types of information.



The Nick Alexander Eight Box Model

Do you know what information you should have, and what information you actually have? That is the question the Eight Box model, originally developed by Nick Alexander, is designed to help. It consists of a four by two matrix shown in Figure 6, with degree of importance on one access, and whether the data changes over time or not on the other. Placing information requirements and actual records into the appropriate box tells you how you need to manage them.

Figure 6: The Nick Alexander Eight Box model, with examples.



	Unchangeable	**Changeable**
Category 1 Legally Mandatory	• Financial Accounts • Safety Incident Reports	• Salary Payments • Invoice Payments
Category 2 Essential to Business Operations	• Product Specifications • Production Plan	• Stock Levels
Category 3 Desirable for Business Operations	• Key Performance Indicator definitions	• Stock Forecast • Performance Measure Values
Category 4 Why keep this information?	• Design drawings for obsolete equipment	• Maintenance records for obsolete equipment

The Case of the Missing Drawing

A major oil company had a refinery on an estuary with an oil terminal on the opposite bank to the refinery and a pipeline between the two. The pipeline was built in the 1920s. In around 1990 the pipeline sprang a leak, which lead to a search for the drawing to say just where the pipeline was. No such drawing was found. The company was eventually fined £1m for, among other things, not maintaining proper records. This led to a check for what records were held to see if any other critical records were missing. Among other records they found a drawing of the kitchen table, for the canteen, that had been demolished five years previously, a poignant juxtaposition to the missing drawing. This incident led to the development of the eight-box model.

Pick a Model for a Situation

Each of the models for classifying information in this section is likely to be helpful in one situation or another when you are considering how to proceed with a particular challenge managing information. So see them as a toolkit you can dip into to pull out the right tool for the job.

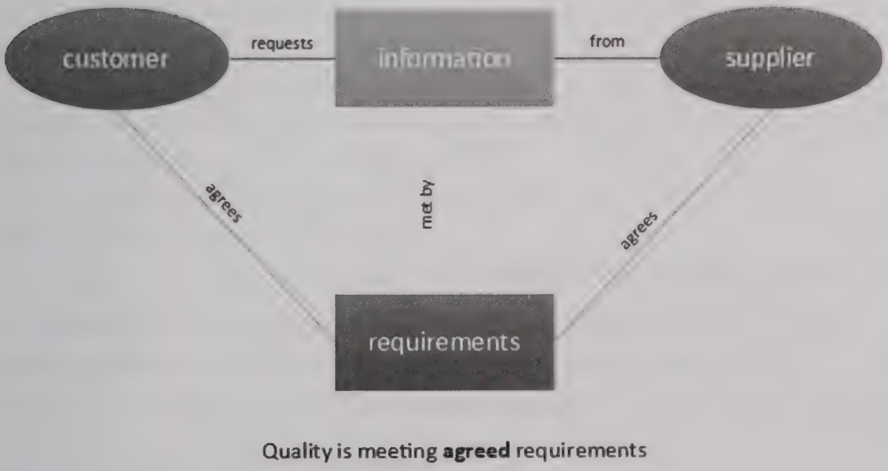
Elements of Information and Knowledge Management

The previous sections have discussed the answer to why one should devote resources to information management. We now outline how one can obtain and manage information with sufficient quality for its purpose.

Information Requirements

One can't fail to meet unstated requirements, yet so often with information, requirements go unstated, and people and organizations are surprised that their requirements are not met, often with considerable cost resulting to rectify the situation.

Figure 7: Requirements need to be agreed between customer and supplier.

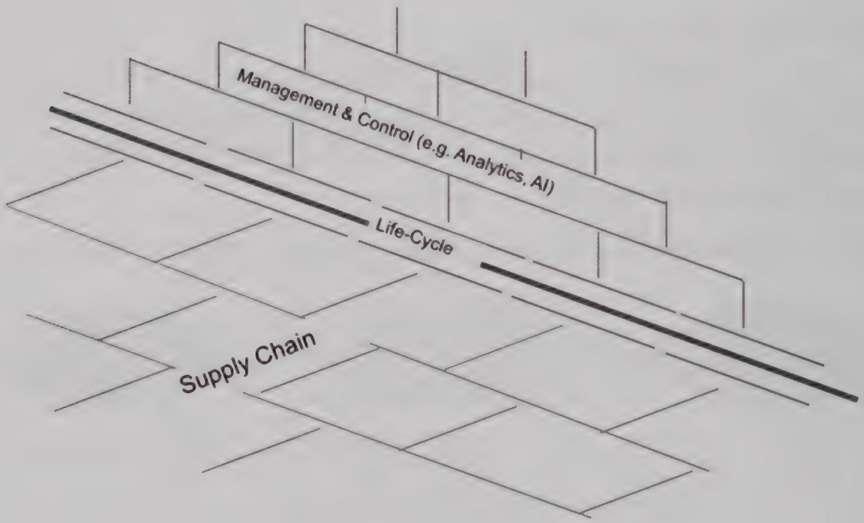


However, establishing requirements requires more than simply stating them, as illustrated in Figure 7, it needs agreement with the supplier of information; in particular about which information, to what accuracy, and when it will be delivered. There are broadly two types of information this applies to:

1. Information about a product or asset, that is essentially part of the product or asset and is necessary for its operation, maintenance, upgrade, and disposal, *e.g.*, its design and performance specification.
2. Information about operations that is used to account for and assess the performance of the business, *e.g.*, sales figures, process measurements.

Requirements for other kinds of information and knowledge such as master and reference data will be derived from these primary requirements.

Figure 8: The dimensions of business and information sharing and interoperability



Data sharing and interoperability

We need to share data when we work with others or for others, and in turn when data comes from multiple sources it must be interoperable. Data sharing and interoperability is one of the hardest things to achieve with data. Some examples of the need to share data are illustrated in Figure 8. This shows that the key dimensions to data sharing and interoperability are:

- Through the lifecycle of an asset, where data that was created, maybe a long time ago, is needed for operations and maintenance, repairs and upgrades, and finally disposal (remember the leaking pipeline).
- Through the management and control of processes where performance data is summarized for analysis across a business.

- Through the supply chain where data associated with products and services needs to be shared between organizations so that it fits with data from other sources.

The key to sharing data is achieving consistency, so that data from different sources fits together. This can be done by building a single data model based on a top-level ontology as set out in West (2011) that is used as a common language to share data.

Information Management

We are now in a position to set out what information management is. Information management is the process that delivers the right information and information that is right, to the right decision-makers, at the right time.

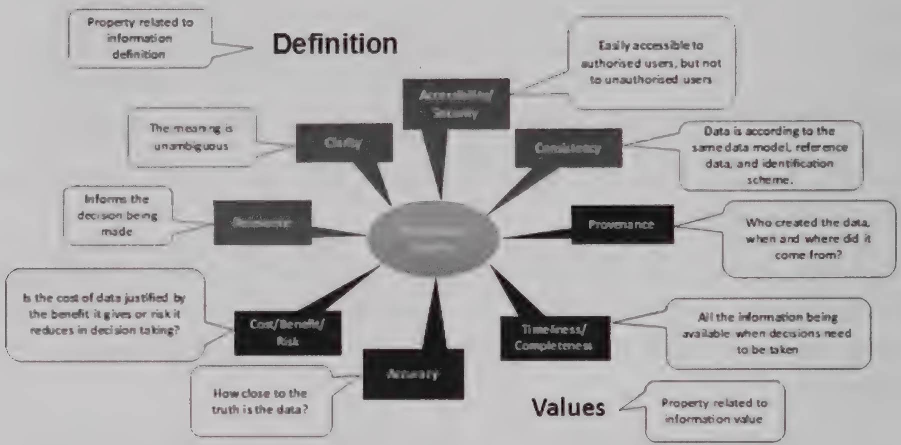
- *The right information* means that it is the information that is required to support decisions in a process, that its meaning is unambiguous, and that it is complete for the purpose and context.
- *Information that is right* means that the process for creating and maintaining the information is defined and followed so that the information is accurate, and consistent, with process performance being measured and processes improved when necessary.
- *For the right decision-makers* means that those who need to have access to the information do have access, and equally, those who should not have access to it do not have access.
- *At the right time* means that the information is available when decisions relying on it need to be taken.

For those who are familiar with quality management, you will by now have noticed that information management is essentially a quality management process. So let's look at information quality management more explicitly.

Critical properties of information

Information has many properties that might be of interest for different purposes. Here we focus on those properties that are critical to its use to inform decisions, as illustrated in Figure 9.

Figure 9: The critical properties of information for making decisions.


Property related to information definition
Definition
Easily accessible to authorised users, but not to unauthorised users
Accessibility/ Security
The meaning is unambiguous
Clarity
Consistency
Data is according to the same data model, reference data, and identification scheme.
Informs the decision being made
Provenance
Who created the data, when and where did it come from?
Is the cost of data justified by the benefit it gives or risk it reduces in decision taking?
Cost/Benefit/ Risk
Timeliness/ Completeness
All the information being available when decisions need to be taken
Accuracy
How close to the truth is the data?
Values
Property related to information value

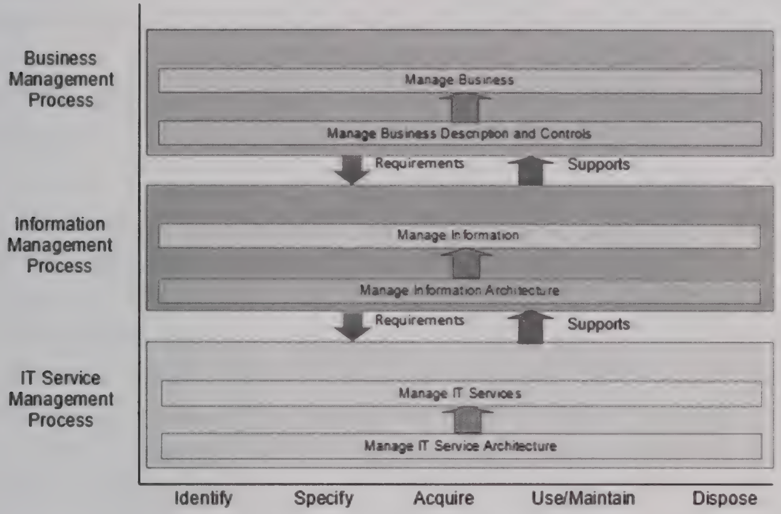
It is these properties that need to be managed. Note the division into properties related to the definition of data – that are determined when an information system is designed or improved, and properties that are about data values, and so are related to when data is created.

Where does information management sit?

It is not unusual to find that information management is missing from organizational processes. When this happens, there can be a disconnect between the business and its information because the linkage is not clear. As a result, information can be seen as a cost that has to be managed, rather than something that provides value. (Lev and Daum, 2004)

Figure 10 shows the proper relationship between the business processes, information management processes, and information service management processes.

Figure 10: The often missing information lifecycle in context.


Business Management Process
Manage Business
Manage Business Description and Controls
Requirements
Supports
Information Management Process
Manage Information
Manage Information Architecture
Requirements
Supports
IT Service Management Process
Manage IT Services
Manage IT Service Architecture
Identify
Specify
Acquire
Use/Maintain
Dispose

- The *business management process* level identifies what the business needs to do to operate in terms of core and support processes, this includes identifying the information required to support decisions in the business process.
- The *information management process* takes the information requirements as input and creates and manages information to support the decisions in the business processes and identifies the requirements for the information services to provide that information.
- The *information service management process* manages services to support the information management requirements. Knowledge graph systems are increasing popular for supporting information service management processes.

Recognizing the importance of the information management process in this way means that there is a clear linkage between business and information services for meeting the information requirements, thereby having clear benefits to the business layer, and then supporting meeting those information requirements with information services giving a clear rationale for their services.

The information management lifecycle

The creation and use of information follows a lifecycle like most things, just with the added complication of different levels of information from Figure 4 to cater for. This is illustrated in Figure 11. The cycle starts with identifying requirements for information or knowledge which then gets detailed into specifying the facts that are to be collected. These facts are acquired as part of a business process, and are validated at their source before being stored. They may be used directly, for example in automation systems, where inferences may be made resulting in actions, such as if the temperature is higher than required, then heating should be reduced. Facts are collected and perhaps concentrated to acquire information which is published where it can be found for use in supporting decisions in a business process. This may cause facts to be updated. The use of information may also give rise to the acquisition of knowledge, which can also be published and used to support decisions, and be transferred and used to support decisions elsewhere as well. Part of the lifecycle management process involves archiving information and knowledge, and when it has reached the end of its useful life, disposing of it.

One useful thing one can do with the information management process is to plot popular topics. Some examples are shown in Figure 12. Although names come and go, the underlying processes were there before, and will continue after, the popular names change, which are often associated with underlying support technologies.

Figure 11: The information management lifecycle.

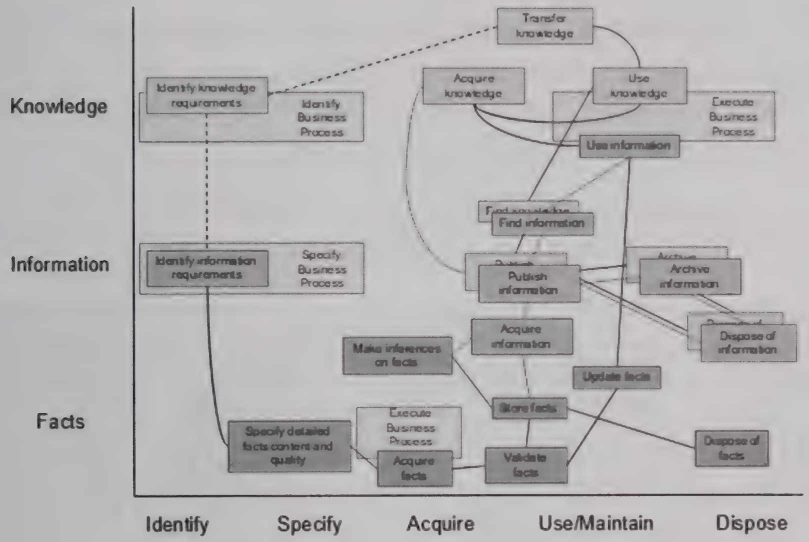
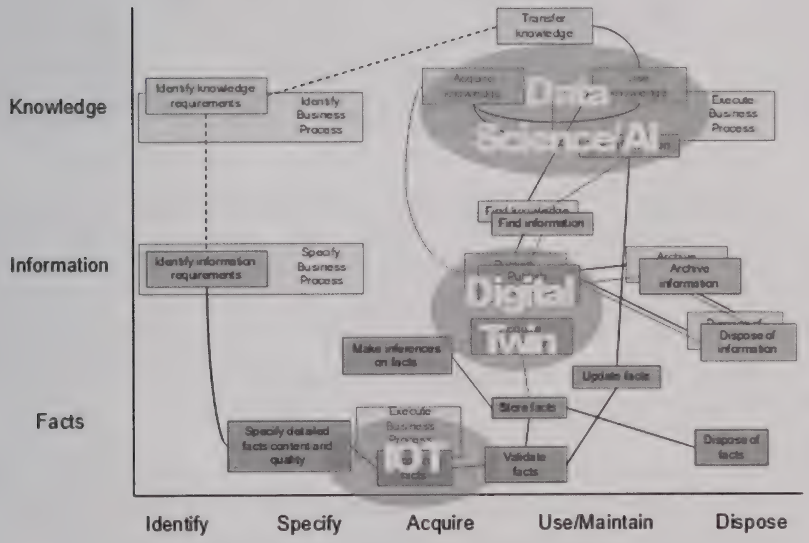


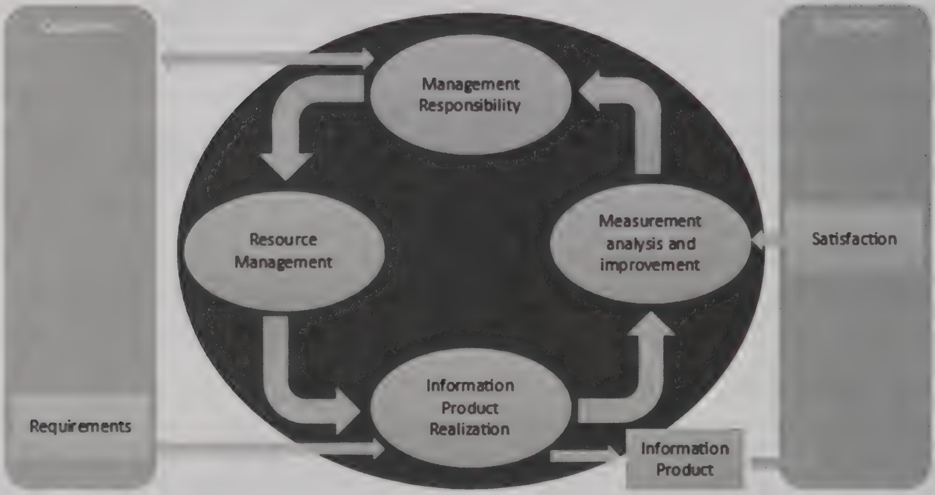
Figure 12: Some popular topics in context.



Applying quality management to the information management lifecycle

We have been talking about information quality, where quality means being fit for purpose to support decision making. However, quality does not happen by itself, it needs positive action such as the quality management process defined by ISO 9001 and illustrated in Figure 13.

Figure 13: Applying the ISO 9001 quality management process to the information management lifecycle.


Management Responsibility
Resource Management
Measurement analysis and improvement
Information Product Realization
Information Product
Requirements
Satisfaction

The first thing to note is that the only change I have made to this from the original at this level is to substitute "information product" for "product". The second thing to note is that the information management lifecycle from Figure 11 is the information product realization process in Figure 13. This is the process that quality management needs to be applied to.

In some places quality management has turned into a tick box exercise with a poor reputation, and I want to emphasize that is not what is involved here. Quality management needs to be a cultural change from a blame culture, where when something goes wrong you look for whose fault it was, to an improvement culture, where when something goes wrong or there is a near miss, it is seen as an opportunity to improve your processes to reduce the chance of recurrence.

If we start at the top in Figure 13 and as broken down in Figure 14, with management responsibility, then this is recognition by the C-suite that information is critical to business success and there is:

- Commitment to managing information and what is needed to achieve that,
- A focus on meeting the needs of the customers for information – those taking decisions,
- Development of corporate policy for information and information management,
- Establishing the responsibilities for data, including setting tasks and targets that can be delegated down the line to those who create and use data, and communicating throughout the organization what is expected,
- Management oversight of information management processes,
- Regular management review of information management progress and performance.

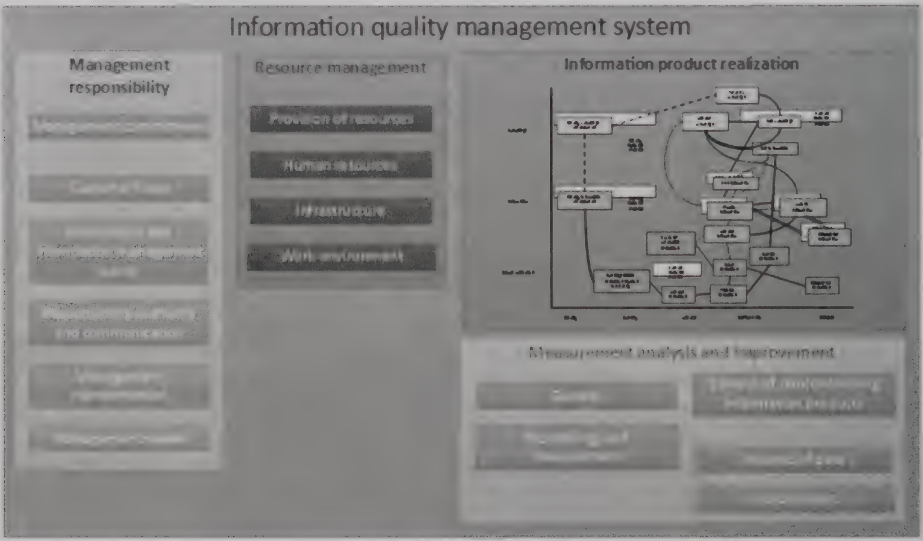
The next key element is providing the resources to do the job. It is surprising how often information management is added as a task to someone who is already fully loaded, and then without tasks and targets being set. It is not hard to see how that will go. Resource management has a number of elements: Human resources, Infrastructure, and Work environment.

As already mentioned, the information product realization process is the information management lifecycle process already described. The key things to note in addition are ensuring the quality requirements of the customers for the information are understood with processes established to ensure they are met, and to check that customers are satisfied with the information product supplied.

Finally, we have the measure, analysis and improvement process. This includes:

- Monitoring and measurement of the information product to ensure that it meets requirements,
- Control of non-conforming information products, by *e.g.,* cleansing,
- Analysis of non-conforming products to identify the root cause of the non-conformity,
- Making improvements to processes to ensure the non-conformities do not recur.

Figure 14: Information quality management process detail


Information quality management system
Management responsibility
Resource management
Information product realization
Provision of resources
Human resources
Infrastructure
Work environment
Measurement analysis and improvement

Establishing an improvement process around your information gives you a way to pull yourself up by your boot laces, systematically improving your performance in information management, and as a result, improving the quality of decision making in your organization.

Conclusion

In this article we have:

1. Shown that information matters because it is used to support taking decisions, so that poor quality information increases the risk of mistakes and disasters, where reducing mistakes increases productivity,
2. Identified that a key enabler of automation is holding information and knowledge as data that can be managed and accessed via services such as those provided by a knowledge graph system.
3. Provided some ways of looking at information that are useful in assessing it and understanding how to manage it,
4. Outlined the elements of information management including,
   - Identifying information requirements,
   - The critical properties of information,
   - The information lifecycle, and
   - Applying quality management to the information lifecycle.

Acknowledgement

This paper is based on work originally done when the author was working for Royal Dutch Shell. This has more recently been adapted and improved through work for the UK Centre for the Protection of National Infrastructure for a forthcoming report “Information Quality Basics”.

References

Baclawski, K. (2021) "A Knowledge Graph Data Model and Query Language", In *J. Wash. Acad. Sci.*, to appear.

Hackitt, J. (2018) "Building a Safer Future", Retrieved on 6 September 2021 from https://bit.ly/3tnXHu8

Lev, B. and Daum, J.H. (2004), "The dominance of intangible assets: consequences for enterprise management and corporate reporting", Measuring Business Excellence, Vol. 8 No. 1, pp. 6-17. https://doi.org/10.1108/13683040410524694

Pressman, R. (2001) Software Engineering: A Practitioner's Approach, Fifth Edition, McGraw-Hill, Retrieved on 6 September 2021 from https://bit.ly/3neJ37d

West, Matthew (2011) Developing High Quality Data Models, Morgan Kaufmann

Issues in Incrementally Adding Better Semantics to Knowledge Graphs

Gary Berg-Cross

RDA/US Advisory Group

Abstract

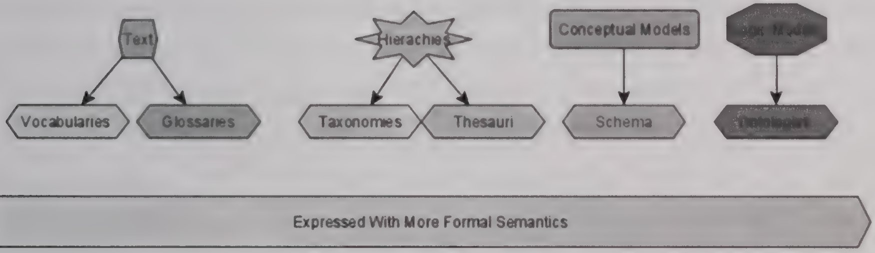
Knowledge graphs (KGs) employ a wide range of semantic resources. However, as is true of complex information systems, harmonizing rich semantic resources requires effort and involves trade-offs. There are practical reasons to start with modest semantics, and then incrementally add enhanced semantic improvements. For this process there are a number of active research projects that are developing light, incremental approaches, methods and tools to support an expanding semantic KG space that has addressed semantic alignment and harmonization. These projects include methods for using existing semantic relations and entities harmonized across controlled vocabularies, glossaries of definitions and ontologies. This article discusses examples of incrementally improving the semantics of less formal schemas that over time is helping to semantically unify richly interconnected heterogeneous data using newly adopted and agreed upon methods.

Introduction

It is common practice to engineer complex information systems using iterative processes to provide incremental improvements (Darrin and Devereux, 2017). This is reflected in principles of extreme programming (XP) which is iterative and incremental and tries to address only a few modeling issues during each phase of work (Choudhari and Suman, 2010; & Dalalah, 2014). A form of extreme programming called the eXtremeDesign (eXD) approach has been used for the development of semantic resources such as knowledge graphs (KGs), ontology design patterns (ODPs), and ontologies. Indeed the start small/iteratively improve strategy of making incremental improvements, starting from some simple semantics, was an implied part of the Semantic Web (SW) vision. It can be seen in the semantic spectrum diagram (Figure 1) that starts with terms and glossaries that have textual definitions and moves to a series of increasingly semantically precise and expressive definitions of conceptual entities that at the far end of the spectrum are represented in some formal language. This original SW vision was that over time more data will be represented in forms processable by computing machines to enable efficient and effective semantic exploration of the Web by bringing "structure to the meaningful content of webpages, creating an environment where software agents can roam across webpages in order to carry out sophisticated tasks for users" (Hogan, 2020). In this vision incremental improvement choices should start at the low end of the spectrum of semantic resources, which might be a single word or phrase and its conceptual definition. At the high end we arrive at a very expressive formal ontology with structured and unambiguous ways of representing domain information. Formal languages used to represent information are intended to specify processable relationships between formally conceptualized data elements and be part of the SW vision, and to employ precise URIs for relationships and properties. Implied in the semantic spectrum model are "bottom-up" engineering approaches leveraging more informal information. This can work, as shown by KG efforts starting with relatively unstructured descriptions of a domain. We can encode initial concepts from words and their implied relations informally understood by domain experts. These concepts can then be simply structured into a prototype model that identifies key concepts. From a simple start increased semantic specificity removes ambiguity and affords a growing degree of sharing and interoperability. The idea of incremental development is attractive since it is a way to use the growing abundance of information and semantic resources hosted on the web and available for applications.

However, in light of the experience with a variety of semantic resources since the original semantic web formulation, there are more than simple, pointwise improvements to be made formalizing web data by say expressing some definition in OWL. In addition to this there are improvements to such related issues as coverage of relevant information, factual or timely correctness, overall structures between items, and methodological questions of how any of these changes may be done efficiently and effectively. To address this I provide examples of both simple, somewhat isolated semantic enhancements, and ones that are broader. I also provide examples of some systematic methods, such as harmonization across a suite of concepts. A challenge of an incremental approach is that while simple concept definitions can serve as links between things on the semantic spectrum, there may be numerous ways of defining a concept. The word "forest" is used conversationally, but there is no universally recognized precise definition. More than 800 definitions of forest are recognizable around the world, including off center relations to forest nurseries and forest roads. To incrementally improve semantics as one moves along the spectrum core relations such type and part-whole relations can be used to reduce ambiguities in a conceptual space. Consistent with this idea early ontological engineering practices aimed at reaching the formal end of the semantic spectrum emphasized that ontology development is necessarily an iterative process with regular revisions, debugging, and progressive deepening. (Noy and McGuinness. 2001). This general process is equally true for KG development.

Figure 1: Semantic Spectrum from Defined Terms to Formally Defined Concepts based on (MA, 2021)



Data abundance provides opportunistic construction of products like KGs by leveraging the large amount of available data, including semantic resources along the semantic spectrum. However, real-world KGs are complex. The data, while vast, includes informal types, is usually incomplete, and is not harmonized. Thus, early phases of work with raw data do not easily reflect a full reality-based model. There are risks to assume that one can develop a fully validated domain-spanning semantic model all at once. As noted by Elsaleh *et al*. (2019), "semantics add further overhead to data delivery, and the processing time to annotate the data with the model can increase the latency and complexity in publishing and querying the annotated data."

For these reasons and others, in practice, most KG projects are largely data driven from the bottom and do not build or use a full and semantically rich, domain ontology. Instead, projects search for low-hanging fruit and tractable semantic resources, such as a centralized, crowd-sourced approach using Wikidata as the foundation or using generalized vocabularies. Still another approach is to leverage a slimmed down ontology (Heller *et al*., 2018). Work often starts with what are called lightweight ontologies (Giunchiglia and Zaihrayeu, 2007). These aid developers (and users) by allowing relatively fast annotation of data with information from glossaries. These are midway along the expressiveness dimension of the semantic spectrum. Lightweight ontologies and related models are largely descriptive and include groups of concepts, concept taxonomies showing sub-classes, and simple, conceptual relationships between concepts. Lightweight ontologies have only a modest number of axioms and relations, and may focus on taxonomic and structuring relations. Heavier or richer ontologies include axioms and constraints beyond hierarchical ones to clarify the intended meaning of the terms involved in a domain. There are a variety of trade-offs between light and heavy models. Obviously, richer axioms done well are closer to domain reality. But a benefit justifying a lightweight approach is to save query processing time when complex relations are involved.

A big advantage of starting with low-hanging resources and lightweight semantics is the desire to make rapid progress. So we typically see that KGs may be implemented using data with limited semantics, such as the use of RDF triples or property graphs with no schema. As a result projects may have masses of data with no consensus on core semantics or what models are reflected in the data. Even in a structured form like RDF, data may be developed from different vocabularies and different perspectives on the data and largely be stored in "dispersed forms in a number of autonomous information silos" (Guizzardi, 2020). As Heflin and James Hendler (2000) put the resulting challenge of integration: "To achieve semantic interoperability systems must be able to exchange data in such a way that the precise meaning of the data is readily accessible and the data itself can be translated by any system into a form that it understands." We need some degree of deep knowledge to support domain reasoning to fulfill this SW vision.

FAIR Guiding Principles

A step towards sound and incremental data management practices regarding new knowledge generation and discovery by individuals and organizations was taken in 2016, with the publication of the 'Findability, Accessibility, Interoperability, and Reuse' (FAIR) Guiding Principles for scientific data management and stewardship (Wilkinson *et al*., 2016). The FAIR principles include enhanced semantics for machine-actionability (*i.e.*, the capacity of computational systems to find, access, interoperate, and reuse data). The "I" in FAIR is concerned with putting machine-readable knowledge on the web, and incremental semantic technology helps to achieve this by developing comprehensible structure and context that makes data easier to reuse and integrate with other data.

The Internet of Things as an example

The Internet of Things (IoT) is one area of interest to KG development that does not yet have a consensus on top-down models or ontologies. This reflects the more general fact that unitary efforts to develop top-level ontologies or even broad and deep domain ontologies have had problems (De, Suparna, Zhou, and Moessner, 2017). Because of the breadth of entities involved in IoT, formal agreements on semantics tend to be avoided to make KG construction quicker. Instead, as noted before, a bottom-up process is often the initial guide. This leverages analysis of implied meanings, as understood by domain experts and/or data analysts, found in structured and linked data. This makes sense since the choice of semantic simplicity avoids complex metadata documentation and encourages faster adoption by end users who can easily grasp the concepts. However, ignoring top-down semantics can sacrifice system quality, making it difficult to interpret query results against intended or reliable agreed upon meaning. For example, without an effective naming authority for Web data, it can happen that different KGs refer to the same thing by different names, and the same names may have quite different meanings (Alexopoulos, 2020). This problem is well known from earlier work on conceptual models of data (Hull, 1996). A modeling example is the difference between a glossary for cars versus the more general one of automotive vehicles, which would contain trucks. Manually resolving these differences can be challenging and involve many trade-offs. Creating a semantic data model is a labor-intensive process, and requires a sound understanding of the selected domains within a KG’s scope along with the relevant ontologies.

Data Heterogeneity Tradeoffs and Challenges

For many reasons, KG building tends to fall back on some idea of incrementally adding semantics as part of later stages of iterative development. However, there are several challenges for improving KG semantics, such as data heterogeneity. Barriers exist to continued, incremental progress, such as finding an easy, effective way to create richer vocabularies, remove ambiguity, and integrate richer semantics into schemas with appropriate constraints and relations. The struggle starts with the problem of conceptual heterogeneity, such as contradictory structures and/or levels in different taxonomies or other lightweight semantic resources. Groups crafting definitions (or building other semantic resources like conceptual models) have varying experience and ability and use different methods. This situation can innocently result in unintended heterogeneity. One may use some knowledge engineering to craft semantically harmonized definitions, but this can be time-consuming, error-prone, and a tedious process. Domain experts can quickly lose interest in such work. Indeed, different domain experts may use varied implicit background knowledge to understand, and later define, concepts with the same name. Experts and knowledge engineers may locate the same targeted concept differently within some conceptual space or hierarchy. In turn, the use of significantly different distinguishing concepts creates conceptualization mismatch in a KG. As a result, the KG may misalign data based on their different underlying source models, even when attributed to hardworking domain experts.

Incremental improvements may also be challenged by the trade-off between coverage and correctness. Coverage is concerned with whether the KG has all the required or desired information. Effectively the answer is always no, in the sense that domain knowledge is indefinitely extensible, and development teams are motivated to look for new ways to provide value to domain users. And it is also true that new sources of data, information and organizational schemes, like language, emerge over time. However, as the coverage increases, the likelihood of a conflict or contradiction also increases. Approaches to the trade-off between coverage and correctness may be addressed differently in particular KGs when difference in the coverage occur and new data sources are introduced. KGs may also need to extend their semantics to describe different realms or regions of the world at the same level of detail and/or from a different perspective. Similarly, differences in granularity occur when we have the same perspective, but at different levels of detail. This occurs, for example, with geographic maps of different scales. Difference in perspective (difference in scope) occurs when two data sources describe the same region of the world, at the same level of detail, but from different perspectives (*e.g.*, a political map vs. a geological map). For all these reasons revisions and modifications in a KG lifecycle may result in ambiguity, redundancy, and modeling inconsistencies that need to be addressed over time.

In general, achieving a base level of interoperable results given the heterogeneity among different information systems is difficult (Maciel, 2017). More than a simple change of representation is needed as part of data integration and harmonization. The simple fact is that in order to achieve a useful degree of interoperability between datasets, either the datasets need to use the same (set of) ontologies, or the ontologies need to be aligned and mapped. How to develop efficient alignment and mapping is one of the areas of improvement emerging from KG research.

Examples of Incremental Improvements

To give the flavor of work that addresses some of the challenges KG and related efforts face, six types of incremental semantics are briefly illustrated, starting with the simple case of improving and harmonizing identifier systems. This seems a relatively simple problem to address, but still exists and is a major concern of the FAIR principles. We then consider four specific ontologies. This section ends with a discussion of automated techniques.

Clarifying Identifier Systems

The wide range of semantic resources often use different identifier systems. For example, the W3C Organization Ontology (ORG) expresses information about organizations, *i.e.*, companies and institutions, including governmental organizations. The focus is on organizational structure (*e.g.*, sub-organizations and classification of these), along with reporting structures (roles) and facility locations (Reynolds, 2014). In contrast, the e-Government Core Vocabularies that were developed in order to provide a minimum level of semantic interoperability for e-Government systems use a different system that covers overlapping concepts with a focus on public services, public organizations, and public services (European Union, 2015; Gerontas. 2020). There are many other examples, and so developing a unified semantic expression of identified semantic resources becomes a more difficult task as more resources are assembled into a KG. In the business domain, the euBusinessGraph ontology represents an example of using a rather specific lightweight semantic model approach to standardize identifier systems within their area of interest. The euBusinessGraph ontology starts by systematically combining and reusing termed concepts from existing ontologies, such as the previously mentioned EU Core Vocabs: W3C Org, as well as others (Roman, 2021). The result is a revised model integrating several semantic resources, such as vocabularies, into a more expressive and detailed model that includes extensions to "application profile, RDF Shapes and data provider mapping documentation." (Roman *et al.*, 2021).

Extensions and Improvements to FOAF

Refinement and extensions of semantic resources to handle expanded requirements are among the typical increments needed to integrate semantic resources. An example of incremental improvement can be seen in the movement from the early conceptualization of Friend of a Friend (FOAF) (Brickley & Miller, 2018). FOAF is a widely used lightweight social network "core vocabulary", but it has vocabulary areas with little or no adequate semantic coverage. This combination becomes a problem as the vocabulary keeps being reused without improvement. The answer is to avoid slavish reuse and to make incremental extensions. Examples of this makes the FOAF case illustrative of incremental development. There are now numerous refinements and extensions for particular areas. This process may start with expansions of the subclasses and with deepening the classes found in early versions of FOAF. Refinements may take the form of defining inclusions between classes and relations, and/or of refining features and restrictions of the relations. A vocabulary example is the extension of the definition of "landform" as "a feature on the Earth's surface that is part of the terrain." to including formative processes (*e.g.*, volcanic process or tectonic movements) or constituents such as magma as part of a volcano system.

A useful example of a social network extension to meet expanded requirements comes from considering new virtual, social relations (El Kassiri and Belouadha, 2017). These include worked out examples covering extensions to social networks in a variety of behavioral-and health-related research areas. These capture more of the rich interactions of social networks (Amit et al., 2020). Enriching FOAF in this way uses the best practice of building on the existing (FOAF) model and illustrates the practice of incorporating ideas from other ontologies. To do this one can reuse one or more lightweight ontologies, adding extra properties and classes as needed. In terms of ontological engineering practice, this reflects adding new competency questions (*i.e.*, the questions a knowledge base can answer) which are not addressed in a current version of a KG or its underlying ontology.

Another FOAF extension example uses ontological imports from the Food and Agriculture Organization's geopolitical ontology (Kim and Viollier 2013). The result is a richer artifact, called FOAF+. It can be used to describe new types of social ties, interactions and new entity features or attributes not included in the earlier, base and lightweight FOAF. Another aspect of incremental improvement is the quality of conceptual, not just representational, expressiveness. Unlike the early idea of the semantic spectrum, one is not just increasing the expressive language around a concept; one has a quality meaning that is prior to the expression of the knowledge. Otherwise, we face, adapting an old expression, the situation of "imperfect modeling information in; imperfect understanding out."

Schema.org and Bioschema

Annotating data for KGs using a lightweight, standard schema is another situation that illustrates semantic enhancement. Schemas represent a target for mapping and are one way to support improved alignments. An example comes from the use of a master data hub like the Schema.org markup vocabulary. Schema.org reflects an effort to standardize lightweight, annotating vocabularies to reduce data heterogeneity, as well as to ensure that websites are more uniformly indexable for search engines and other web services (Guha *et al*., 2016). As previously discussed, KGs typically start with the low-hanging fruit available from annotations that can be extracted from websites. An incremental improvement is to standardize these by verifying them against the relevant domain part of the Schema.org vocabulary.

Extensions of Schema.org for the bio-science realm illustrate some of the progress. For example, extensions to Schema.org as part of Bioschema work (Franck. 2018) allows searching for and finding data about specific biological entities (*e.g*., particular genes, proteins, and taxa). This enhancement uses entity profiles. An example of this is the idea of TaxonName. This is used to specifically annotate taxonomic name registries.

Guidelines for use are provided that describe how to leverage existing vocabularies such as Darwin Core or Wikidata. Bioschema can be used to improve biology oriented KGs by enabling semantic cross-linking between any KG that extracts data from Bioschema marked-up websites.

The Bioschema enhancement also follows FAIR principles. It includes semantics to help discover data repositories storing experimental results, along with the storage location of specific biological samples. To support this, relevant controlled vocabulary terms drawn from existing ontologies have been imported. A cited example is the protein profile which requires a unique identifier, and recommends listing transcribed genes and associated diseases. For proteins Bioschema points to recommended terms from the Protein Ontology and Semantic Science Integrated Ontology.

As an incremental improvement to Schema.org, Bioschema specifications go beyond simply adding new types and properties for biological entities. It includes more structure, providing constraints on the Schema.org model. These constraints capture and require things like minimal information properties agreed by the bio-community. These fall into categories of mandatory (M), recommended (R), or optional (O). Another enhancement is that the cardinality/occurrences of properties have been added.

Success with semantic enrichment such as Schema.org generates support from tools to automate some of the activity. As an example, a semantic validator has been developed to help ensure the syntactic correctness and completeness of the annotations from a Schema.org perspective (Panasiuk *et al*., 2019).

Sensor, Observation, Sample, and Actuator ontology (SOSA) an Example of a Design Pattern Approach

SOSA is a lightweight but a general-purpose pattern-based specification for modeling the interaction between the entities involved in the acts of observation, actuation (actions triggered by observations), and sampling. As an incremental example, SOSA is the result of rethinking the W3C-XG Semantic Sensor Network (SSN) ontology. It reflects changes in scope and target audience, technical developments, as well as lessons learned over the past years. SOSA, as well as its base SSN, are examples of an incremental approach to semantics using ontology design patterns (ODPs)

(Gangemi, 2005). ODPs help address the reuse problem of complex semantic resources, such as an ontology like DOLCE, where only certain “useful pieces" of a comprehensive (foundational) ontology, may be of interest to a KG. This is based on the observation that the cost of reuse from a large, but shallow ontology, may be a higher cost on resources than developing from scratch a scoped ontology for particular purposes. ODPs can serve as a step towards more structured and semantically rich metadata, even to extend something like Schema.org markups. ODPs reflect the understanding that often to practically solve semantic problems, it is productive to agree on minimal requirements imposed on a relevant family of concepts (Kuhn, W. 2009). ODPs (aka microtheories) represent small, well engineered, coherent, minimally constrained schemas. Light, general ontology patterns function as a modular consistent core that can serve as a starter set from which more varied and detailed models or, as needed, larger ontologies can be built. In effect ODPs serve as an initial constraining network of “concepts” within a common framework using vocabularies allowing people to incrementally extend and align them for various purposes. The overall impact to support some degree of common interoperability, including easy data sharing via a KG using one or more ODPs.

The eXD methodology, mentioned earlier, uses an agile approach to ontology engineering and focuses on the reuse of ODPs (Presutti *et al*., 2009). Among ODP best practice qualities that enable incremental semantics are explicit documentation of design rationales, and the use of best re-engineering practices to facilitate reuse. It is also worth noting that a KG design around a family of ODPs can provide a very concise but informative view of the overall content of a KG (Asprino *et al*., 2021).

ODPs like SOSA support a progression from the light semantics of something like FOAF to heavier semantics that helps avoid legacy and silo building problems (Janowicz *et al*., 2019). As an example of how ODPs afford easy improvement, SOSA uses axiomatization but does not make any formal restrictions on how to use observable properties. This allows for alternative observational models. Observed features of interest, for example, are not restricted to objects like roads. They also can include events such as rush hour traffic. As with other semantic products, over time there has been a recognition for refinement in ODPs. In the case of SSN, which evolved into SOSA, this process included recognizing the need for a central concept of observational sampling over time. Another recognized improvement was that observations may not be carried out on the entire feature, but on samples of a feature and/or as part of sensing some spatiotemporal region that serves as a proxy for a feature (Taylor *et al.*, 2019).

Enhancing Earth Sciences Ontolgoies: adding EnvO axioms to SWEET

A fifth illustration of incremental semantic improvement comes from experiences with broad ontologies like the Semantic Web for Earth and Environmental Terminology (SWEET). SWEET is a lightweight ontology with broad coverage, but sporadic definitions that historically served as a starting point for concepts within the Earth Sciences (DiGiuseppe, Pouchard, and Noy. 2014). Often, richer semantics were added for particular domains. A more semantically richer, but topically overlapping, ontology is EnvO, the Environmental Ontology. EnvO includes semantically controlled descriptions of environmental entities and rich axioms. It thus serves as a quality semantic resource for research and is widely cited. For example, the Darwin Core glossary uses EnvO for habitat descriptions. Over its life, EnvO has been continually extended beyond its initial goal to represent biomes, environmental features, and environmental materials pertinent to genomic and microbiome-related investigations. The need for environmental semantics is common to a multitude of fields, and thus EnvO's use has steadily grown since its initial description. Its scope has expanded, been enhanced, and generalized, so the ontology can support its increasingly diverse applications, as shown by the range of updates in a recent release (EnvO, 2021). One notable example of a recent extension is as a semantic resource for Cryosphere concepts (Berg-Cross and Vardemann, 2020). Work on a common Cryosphere model to be added to EnvO actually started with a 14-hour hackathon on glaciers (Glacier Hackathon, 2019). The session leveraged and harmonized a focused portion of a rich collection of definitions developed by the World Meteorological Organization's (WMO) Global Cryosphere Watch (GCW). This GCW work expanded on prior work on sea ice ontologies (Duerr *et al.*, 2015) and had recently collected some 27 cryospheric glossaries containing a total of 4147 terms. Importantly, semantic analysis showed that only 2249 were unique and terms could be organized into useful categories; namely, those

- that were well-formed and documented and not problematic from a semantic standpoint,
- where multiple definitions could be coalesced into a single definition, and
- where the terminology was inconsistent and therefore problematic from a semantic standpoint, and where community resolution was needed to either agree on a definition or to split the terms up into separate entities, *etc*.

The subsequent hackathon's objective was to develop a refined conceptual model organizing relevant terms into glacial object types, features, composition (*e.g.*, frozen water matter) and processes. The overall hackathon experience of building a conceptual model with some ODP structures was successful enough for participants to seek a way to continue this work on a regular basis. A goal was to use this as a way of aligning and enhancing the SWEET and EnvO ontologies. This was an important test case because both ontologies were independently developed, but both ontologies contain many of the same important semantic resources of the environmental and earth science (ESS) domain. Discussions had been underway about how to align portions of them. A domain like the cryosphere with some harmonization efforts completed presented a good test area for ontology alignment and enrichment. To support this, a Semantics Harmonization cluster was formed with the Earth Science Information Partners (ESIP) Semantic Technology Group to develop a harmonized cryospheric glossary leveraging cryospheric terms in both Envo and SWEET.

As a start, GCW's harmonized definitions were used to add content to the EnvO while simultaneously mapping the results to existing classes in the SWEET Ontology. EnvO's ontology is more richly axiomatized than SWEET since it is part of the Open Biological and Biomedical Ontologies (OBO) Library, and employs recommended practices and technologies for developing expressive and interoperable ontologies in OWL. As ontologies like EnvO and SWEET are enhanced, they in turn can support their domains for KG development.

Below is an example of a semantic update to the EnvO description and axioms for the concept of "ice shelf":

- An ice shelf is an ice mass attached to the coast

- An ice shelf is at least 2 meters in thickness
- An ice shelf forms where a glacier or ice mass flows down to a coastline and onto the ocean surface and
- An ice shelf grows by annual snow accumulation or by the seaward extension of land glaciers.

Some corresponding Envo Axioms are:

- partially surrounded by some atmosphere
- attached to some sea coast
- has quality some buoyancy
- adjacent to some marine water body
- formed as result of some snowfall
- a land ice mass
- formed as result of some mass ice flow

As a whole, SWEET remains less axiomatized than EnvO, but has been updated to use the same harmonized definition that ENVO does. And to support ontology alignment, axioms have been added to SWEET asserting such things as a "closeMatch to EnvO Ice shelf". This allows cross-fertilization with SWEET's long-standing usage in the Earth and environment domain, and offers a pathway for its incremental development.

Relations like "closeMatch" expressed in RDF are part of the lightweight SKOS standard. They provide only modest expressivity for mappings such as mentioned above to relate terms in EnvO and SWEET. A popular formalism for relating terms is SKOS, which uses RDF to provide some formalization of various types of controlled vocabulary. These include classification schemes, subject heading lists, and taxonomies. This promises some degree of automation for finding relevant terms and for aligning similar terms. But using SKOS to define vocabularies and term relations can lead to problems down the road. Since it has semantic limitations, some of its modeling is suggestive, rather than constraining enough to reduce ambiguity. For example, SKOS doesn't provide axioms to explain the similarity and work can struggle to upgrade the SKOS model into a more expressive language like OWL. There is no straightforward path between the two languages without conceptual analysis (Jupp, Bechhofer, and Stevens, 2008). Additional incremental improvements can address this using knowledge engineering practices.

Ontology and Knowledge Learning

The previously noted enormous increase in data, both structured and unstructured, available on the web and in data silos, represents a great opportunity for KG building efforts. But there are known difficulties with some associated data management tasks that have traditionally been done with some degree of handcrafting. However, handcrafting big ontologies and other semantic resources like KGs remains a time-consuming, difficult task. This fact ensures that automated or semi-automated acquisition of ontology from text and structured data remains an active research area with a big payoff potential for populating KGs or building ontologies. Semantic resources like categories of nouns and taxonomies can be learned from texts and even enhanced using automation and machine learning (ML). One way to do this is to extract knowledge from resources on the low end of the semantic spectrum. This idea has been understood for a while (Faure, Nédellec, &, Rouveirol,1998), and efforts have been used dividing the learning process and supporting automation into four different phases: extract concepts, prune, refine, and import or reuse concepts (Mädche, 2005). But the variability of free text can cause a high error rate if automation is based on shallow natural language processing.

While progress has been slow, there have been signs of progress. Inspired by the SW idea of a spectrum of resources. A variety of products can be learned by using a combination of association rules, formal concept analysis, and clustering. Some outputs relevant to KGs focusing on ontologies are shown in Figure 2.

The layer cake model in Figure 2 shows a multistage learning process, starting with terms that are the most basic building block for knowledge learning. Extracted terms can be used to feed into a higher stage using glossaries and thesauri that come with descriptions, definitions and some relations from verb phrases. This process, in turn, provides a basis to define concepts and for generating hierarchies. Along the way, nouns and noun phrases that have the same relations become synonym candidates. All of this helps build a KG information structure that can address questions like “What are the characterizing words, nouns, verbs, and adjectives typically used in this domain?” Ontology learning methods thus provide a way, noisy as it is, to start defining domain knowledge for a KG given a domain glossary, and it can be more than a simple translation (Bozzato, Ferrari, and Trombetta, 2008).

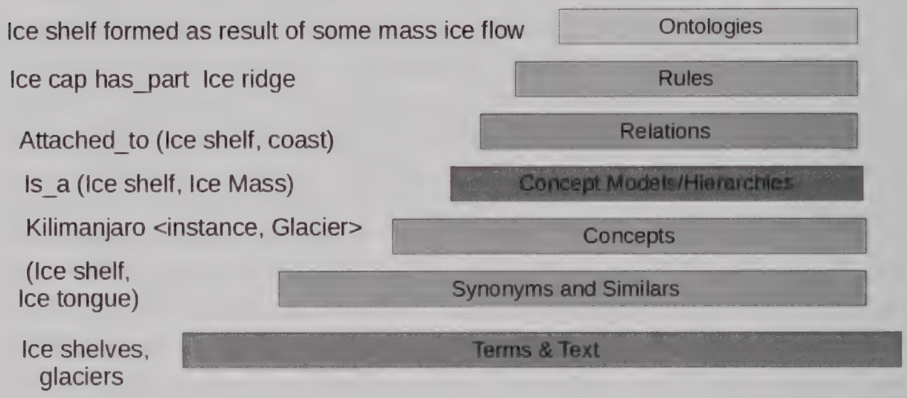

Ice shelf formed as result of some mass ice flow
Ontologies
Ice cap has_part Ice ridge
Rules
Attached_to (Ice shelf, coast)
Relations
Is_a (Ice shelf, Ice Mass)
Concept Models/Hierarchies
Kilimanjaro <instance, Glacier>
Concepts
(Ice shelf, Ice tongue)
Synonyms and Similars
Ice shelves, glaciers
Terms & Text

Figure 2: Ontology Learning Layer Cake Hierarchy based on (Buitelaar, Cimiano, and Magnini, 2005

Still, the typical automatically learned semantic resources need humans in the loop. Auto-generated products need to be inspected, validated and modified by domain experts and knowledge engineers before they can be a basis for being accepted, formalized in an ontology, or applied by an application like a KG. Currently, the practical emphasis of such efforts is not to create a final, perfect ontology for a particular domain, but to create reasonable vocabularies first and then incrementally create schema patterns with domain terms and definitions that are good enough (*i.e.*, constrained and defined enough) for people to start using them for publishing data on the Web. A goal is also to support data integration. As with work with different vocabularies and models, a large part of the effort is about harmonizing things to make products useful. And for that we need interdisciplinary teams (Berg-Cross, 2015).

Useful Practices and Some Guiding Principles

Based on varieties of experiences, some improvements in ontological engineering may be suggested to incrementally advance and harmonize semantic resources. The previous examples suggest some ideas, starting with the need to provide unique, persistent and consistent entity identifiers that are computer processable, but the identifiers should also be relatable and sensible to humans. At a minimum, following FAIR principles, this is needed for findability, but is supplemented when semantic resources are registered or indexed in a searchable resource such as in EnvO. The need for data to be described with rich metadata is also featured in FAIR principles. An incremental approach fits into FAIR's principles for developing and publishing digital material. The use of community standard vocabularies for data should follow FAIR, provide quality, harmonized definitions and associated managed schemas. Vocabularies need to have reliable governance and organizational commitment to FAIR principles and associated ideas like commitment to linked data principles, stable IRI's and associated sensible funding. These are important, as is assessing the readiness of terms, definitions and concepts for use as semantic resources. Incremental improvement efforts need to assess the quality of resources like community accepted glossaries that may exist. Work should leverage and reuse structured data and vocabularies as much as possible. As noted in the work on Cryosphere, progress was enabled by starting with existing vocabularies that were already studied and had some harmonization underway.

Making domain assumptions explicit over time is an important enhancement as we move up the semantic spectrum. References to a foundational ontology, such as done in the OBO Foundry, is an example (Smith et al., 2007). Community efforts like the OBO Foundry represent other basic practices and include principles that are simple, yet important, such as ensuring a stable URI for each refined concept. This goes beyond URL mintings made by individual projects. Efforts like the OBO Foundry involve a community commitment to maintaining a place for managed concepts that assures access in perpetuity.

Broadly, methodologies like eXD or UPON Lite represent useful starting points for projects. The UPON Lite methodology is notable in that it supports the rapid prototyping of trial ontologies that can be extended more easily by enhancing the role of domain experts (using spreadsheets) rather than the constant need for knowledge engineering experts (De Nicola, and Missikoff, 2016). The general advice is to address one modeling issue at a time. For KG development, a useful guideline concerns re-engineering ontological patterns using transformation rules. These can be applied to create a new, target ontology starting from elements of a source model (Blomqvist, Hammar, and Presutti, 2016).

Guidelines also exist for refactoring an ODP. Some concepts may be defined as a hybrid or mosaic of ideas containing a mix of more fundamental concepts. Such marbled concepts have to be decomposed into more rationalized component parts or subtypes since termed concepts should be orthogonal. A guiding principle is that if overlap exists among termed concepts, then there is more than one concept in play. In addition to making use clearer, refactoring affords an opportunity to combine distinct concepts in useful ways. The eXD method provides rules to transform an existing ontological piece, say expressed in OWL DL, due to a requirement change. A typical example relevant to KG development is when a KG is initially populated with individual instances and then advances the organizational use of class structures. Another form of enhancement is moving from object properties to classes (Kasri and Fouzia, 2016).

It is worth noting that ontological methods like eXD require community involvement in the form of interdisciplinary teams. A central emphasis is on the need for domain expertise to address the quality curation that is needed for incremental improvements. This often starts with identifying who are the interested parties needed to improve some collection of semantic resources for some purpose. A typical driver is the need for better data interoperability within a domain. As a bonus along the way, domain experts may provide a seed definition that can expand into new areas of work.

As semantic enhancements are considered, some very general concepts about quality apply to semantic resources across the spectrum, including those such as KGs that make use of many if not all parts of the semantic spectrum. They should not be traded off without thought as we move from one level to another, or as part of efforts to harmonize resources across the spectrum. These concepts are similar to criteria that judge the quality of ontologies and follow from Gruber's (1995) original list including Clarity, Coherence, Extendibility and Minimal encoding. Gómez-Pérez (1996) suggested related criteria that overlap the first three of Gruber's criteria; namely, Consistency and Conciseness (*i.e.*, definitions need clarity and should minimize ambiguity by being expressive in few words). Gómez-Pérez adds that definitions should be complete is some sense and that over time any revisions should capture some core essence of what is known about the real world as part of some finite structure and system. Gómez-Pérez also adds the idea of Definitional Sensitiveness; that is, a definition's core should be stable in the face of small changes. Some of these qualities are expanded as follows:

a. Clarity: the concepts in a semantic resource should be defined in a formal way that communicates the intended meaning of defined terms as understood by a domain community. Definitions should be brief, with objective necessary and sufficient conditions. If the scope of a defined concept is changing and becoming more formal, that should be documented.

b. Coherence: concept definitions, especially the formal aspects, should stand up to rational analysis. For example logical inferences should make sense and be consistent with the overall domain understanding. It follows the advice – "first do no damage." Wikipedia concept entries have often been the start of a knowledge base. However, based on experience, we know there are consistency questions about these meanings since Wikipedia represents a loosely governed heap of diverse material. While Wikipedia remains a source to start with, it is often better to consider controlled sources that can supplemented by assembling definitions and actually analyzing terms from relevant domain vocabularies which are likely to be stable. This was the case with work on a Cryo vocabulary. A concept does not stand alone, and its definitions and inferences should make sense in light of the definitions and inferences of related concepts. Therefore, methods are needed to reach agreement on conceptualization across concepts. A starting point can be an agreement that something like an ontology or ODP can be assembled by anchoring their concepts to a local, harmonized glossary of terms and a set of agreed upon relationships between them. From this one may consider cross-domain integrating or bridging concepts that meaningfully relate concepts between domains as needed for a wide-spanning KG.

c. Extensibility/Scalability: a suite of semantic resources such as a glossary, KG, ODP or ontology should design in and anticipate expansions and extensions to address likely, new requirements or as semantic resources are added. This is a natural part of a change management process. For semantic enhancements this includes consideration of how to scope a domain, what parts of definitions can be axiomatized and, where possible, use formulations in existing quality ODPs and ontologies for this. Any improved knowledge should are captured in competency questions as laid out in the eXD method. The previously mentioned SOSA pattern is built on an extendable vocabulary that can be combined with other related ontologies or ODPs, such as SSN, to provide a more rigorous axiomatization where needed. Extensibility is promoted by use of such ODPs as a small conceptual foundation with some general concepts that are useful. Examples include "physical-object", "event", "process" and "situation". These can be used as foundations as a step to a systematic foundry and can be adapted over time to work across a range of certain tasks. A good example of a useful pattern is the formalization of Winston's Taxonomy Of Part-Whole Relations for use in ontologies (Shimizu, Hitzler, and Paul, 2018). A guiding quality related to extensibility that is seen in foundational ODPs is that of "minimal semantic commitment" which states that a semantic resource should require the minimal ontological commitment in order adequately to support any anticipated knowledge sharing activities.

d. Coverage: Coverage asks the scoping or completeness question, "Does the graph have all the required information?" Obviously this is a matter of degree, since even lightly formalized knowledge can't practically provide full coverage of a domain or a suite of domains. As with the work to harmonize Cryo definitions, some scope must be considered. Furthermore, even as extensions and refinements are applied, we know that a KG or an ontology is an approximation and not fully correct. This reflects the reality of a trade-off needed between coverage and correctness. Where and how this trade off occurs is likely to be different in each KG (Paulheim, Heiko. 2017).

On a more detailed, lower level, there are good semantic resource management practices to consider such as:

a. Tracking new concepts added into a KG or ontology, and documenting these incremental changes
b. Versioning concept relationships so that interested parties can explore how they have changed over time
c. The need for Memoranda of Understanding (MOU). As with FAIR, explicit group agreement on meaning and commitment to a core family of defined terms that are central to much work. On a practical level, some between group MOUs on this may be helpful.

d. Nascent technologies exist to help control the quality of semantic enticements, especially to ontologies. These include tools to help with alignment between and among semantic resources such as ROBOT (Jackson *et al.*. 2019) which is a generic command-line tool using a Java library. ROBOT performs common ontology and KG supporting chainable tasks including: computing differences between OWL ontology versions, merging, extracting OWL modules, reasoning, and some support for "explanations". A ROBOT template has been adapted to help align SWEET and EnvO as part of the Cryosphere harmonization previously described. There are also tool suites for broad work on semantic resources. For example, the suite of web-based tools that are part of Ontoanimal (*e.g.*, Ontofoxp for reuse of terms, Ontorat to edit existing terms, and Ontobeep for ontology comparisons) support iterative, extensible ontology development (He *et al.*, 2018). Boomer (Mungall *et al.*, 2016.) is another helpful semantic tool. It uses a combined logical and probabilistic approach to translate mappings into logical axioms for merging ontologies. Tools and practices for pattern-based modular ontology engineering have also been developed (Shimizu, Hammar, and Hitzler, 2020). These reflect a portion of the leveraging and promoting of sociotechnical practices as part of a strategy enabling incremental semantics. Light, rapid methodologies, like the UPON lite ontology approach (De Nicola, and Missikoff.2016) can help make the ontologies more available for KG structuring and consider a range of semantic resources growing from domain terminologies to domain glossaries, taxonomies and simple axiomatized relations.

Conclusions

Knowledge graphs employ a wide range of semantic resources, and bringing them together with rich semantics remains a challenge. But as we have seen, there are now good practices and active research areas using incremental semantic improvements to support this rapidly expanding space of KGs. These range from very focused enhancements to representation of a concept to alignment as part of a community schema or agreed upon domain conceptualization or as part of a larger process to harmonize related but disparate domain vocabularies and assimilate these into extant ontologies. We can expect more systematic development to bring various parts of incremental methods together. These could be evidenced in improvements targeted to ontological engineering methods for alignment and harmonization. As an example, we can expect to see how to use existing semantic relations found in quality ontologies like EnvO to help improve formal definitions from glossary sources. In turn harmonized glossary definitions provide a rich source of material for assimilation into ontologies and for ODP formation. The result over time will be to help semantically unify richly interconnected heterogeneous data using community adopted and agreed upon methods.

Among the targets for continued research is the use of NLP and ML-based extraction from definitions as seeds for ontology improvement and development of better ML automation. Applying ML to build lightweight ontologies is still exploratory, but promising (Wong, 2009) and there still are challenge of learning non-taxonomic relations needed for ontologies from text. But ML is starting to support tasks such as term extraction, conceptualization and enrichment using reverse engineering, schema mapping, data mining, and ML. Enhanced semantics may also follow from work on ontological subsumption, mapping and ontology matching and handling similarity and analogy models, as well as bottom-up semantics from ML approaches and their symbolic ontology models.

Hopefully, technical and user centered methodological improvements will lead to more community input and involvement. Community involvement is necessary to guide the development and refinement of any semantic resource, including KGs. A core group may offer starting points and facilitate development, but the conceptualization of a domain along with its vocabulary belongs to the community of domain and interdisciplinary experts working with knowledge engineers. This is needed to show how extracted concepts can be grouped, related, and subdivided according to their human-understood semantics in context.

Future extensions to look forward to as part of revised and agreed upon methods include those needed to harmonize and axiomatize entity definitions and extensions using the best ontological engineering practices. But it is likely that we will rely on human agreements for a long time, using devices and helpful artifacts such as community schemas and consistent patterns for resource alignment.

References

Alexopoulos, Panos. *Semantic Modeling for Data.* O'Reilly Media, 2020.

Amith, Muhammad *et al*. "Friend of a Friend with Benefits ontology (FOAF+): extending a social network ontology for public health." *BMC Medical Informatics and Decision Making* 20.10 (2020): 1-14.

Asprino, Luigi *et al*. "Pattern-based Visualization of Knowledge Graphs." *arXiv preprint arXiv:2106.12857* (2021).

Berg-Cross, G. "The GeoVoCamp Workshop Experience and Ontology Design Pattern Development." in Narock, T., and P. Fox. Eds. *The Semantic Web in Earth and Space Science. Current Status and Future Directions* 20 (2015): 171.

Berg-Cross, G and Vardemann, C. Semantic Harmonization: Illustrating Harmonization Levels, Winter ESIP Meeting, Januarym 2020.,

Brickley D, Miller L. FOAF (Friend of a Friend); 2000. http://www.foaf-project.org/. Accessed 28 June 2021.

L. Bozzato, M. Ferrari, and A. Trombetta. Building a domain ontology from glossaries: a general methodology. In A. Gangemi, J. Keizer, V. Presutti, and H. Stoermer, editors, SemanticWeb Applications and Perspectives, SWAP 2008, volume 426 of CEUR Proceedings, 2008.

Buitelaar, Paul, Philipp Cimiano, and Bernardo Magnini. "Ontology learning from text: An overview." *Ontology learning from text: Methods, evaluation and applications* 123 (2005).

Buitelaar, Paul: Overview *"Ontology Learning - Some Advances"* http://ontologforum.org/index.php/ConferenceCall_2017_03_01

Blomqvist, Eva, Karl Hammar, and Valentina Presutti. "Engineering Ontologies with Patterns-The eXtreme Design Methodology." *Ontology Engineering with Ontology Design Patterns* 25 (2016): 23-50.

Choudhari, J. and Suman, U. 2010. Iterative Maintenance Life cycle Using eXtreme Programming. In Proceedings of the International Conference on Advances in Recent Technologies in Communication and Computing (Kottyam, India, October 15 - 16, 2010). ARTCom-2010. IEEE Computer Society, 401 – 403

Dalalah, Ahmad. "Extreme Programming: Strengths and Weaknesses." *Computer Technology and Application* 5.1 (2014).

Darrin, M.A.G. and Devereux, W.S., 2017, April. The Agile Manifesto, design thinking and systems engineering. In *2017 Annual IEEE International Systems Conference (SysCon)* (pp. 1-5). IEEE.

De Nicola, Antonio, and Michele Missikoff. "A lightweight methodology for rapid ontology engineering." *Communications of the ACM* 59.3 (2016): 79-86.

De, Suparna, Yuchao Zhou, and Klaus Moessner. "Ontologies and context modeling for the Web of Things." *Managing the Web of Things* (2017): 3-36.

DiGiuseppe, Nicholas, Line C. Pouchard, and Natalya F. Noy. "SWEET ontology coverage for earth system sciences." *Earth Science Informatics* 7.4 (2014): 249-264.

Duerr, Ruth E. *et al*. "Formalizing the semantics of sea ice." *Earth Science Informatics* 8.1 (2015): 51-62.

El Kassiri, Asmae, and Fatima-Zahra Belouadha. "A FOAF ontology extension to meet online social networks presentation and analysis." *2017 IEEE International Conference on Power, Control, Signals and Instrumentation Engineering (ICPCSI)*. IEEE, 2017.

EnvO May, 19, 20921 release. https://github.com/EnvironmentOntology/envo/releases

European Union, e-Government Core Vocabularies handbook, doi:10.2799/97439, https://ec.europa.eu/isa2/sites/isa/files/e-government_core_vocabularies_handbook_0.pdf (2015)

Faure, David, Claire Nédellec, and Céline Rouveirol. "Acquisition of Semantic Knowledge using Machine learning methods: The System" ASIUM"." *Universite Paris Sud*. 1998.

Michel, Franck. "Bioschemas & Schema. org: a lightweight semantic layer for life sciences websites." *Biodiversity Information Science and Standards* 2 (2018): e25836.

Gangemi. A.Ontology design patterns for semantic web content. In Proceedings of the Fourth International Semantic Web Conference (ISWC-05), 2005.

Gerontas, Alexandros. "Towards an e-Government semantic interoperability assessment framework." *Proceedings of the 13th International Conference on Theory and Practice of Electronic Governance*. 2020.

Giunchiglia, Fausto, and Ilya Zaihrayeu. "Lightweight ontologies." (2007).

Glacier Hackahon, 2019, https://github.com/Vocamp/Virtual-Hackahon-on-Glacier-topic#readme

Gómez-Pérez, A. Towards a framework to verify knowledge sharing technology. Expert Syst. Appl. 1996, 11, 519–529.

Gruber, T.R. Toward principles for the design of ontologies used for knowledge sharing? Int. J. Hum. Comput. Stud. 1995, 43,907–928.

Guha, R.V., Brickley, D., Macbeth, S.: Big data makes common schemas even more necessary. CACM 59 (2) (2016), http://dx.doi.org/10.1145/2844544

Haller, A., Janowicz, K., Cox, S., Le Phuoc, D., Taylor, K., Lefrançois, M.: 'Semantic Sensor Network Ontology', https://www.w3.org/TR/vocabssn/#intro, accessed January 2018

He, Yongqun *et al*. "The eXtensible ontology development (XOD) principles and tool implementation to support ontology interoperability." *Journal of biomedical semantics* 9.1 (2018): 1-10.

Heflin, Jeff, and James Hendler. *Semantic interoperability on the web*. MARYLAND UNIV COLLEGE PARK DEPT OF COMPUTER SCIENCE, 2000.

Hogan, Aidan. "The semantic web: Two decades on." *Semantic Web* 11.1 (2020): 169-185.

Hull, Richard. "Managing semantic heterogeneity in databases: a theoretical prospective." Proceedings of the sixteenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems. 1997.

Janowicz, Krzysztof *et al*. "SOSA: A lightweight ontology for sensors, observations, samples, and actuators." *Journal of Web Semantics* 56 (2019): 1-10.

Jackson, Rebecca C. *et al*. "ROBOT: a tool for automating ontology workflows." *BMC bioinformatics* 20.1 (2019): 1-10.

Jupp, Simon, Sean Bechhofer, and Robert Stevens. "SKOS with OWL: Don't be Full-ish!." *OWLED*. Vol. 432. 2008.

Kasri, Soumaya, and Fouzia Benchikha. "Refactoring ontologies using design patterns and relational concepts analysis to integrate views: the case of tourism." *International Journal of Metadata, Semantics and Ontologies* 11.4 (2016): 243-263.

Kim S, Iglesias-Sucasas M, Viollier V. The FAO geopolitical ontology: a reference for country-based information. *J Agric Food Inf.* 2013;**14**(1):50–65.

Kuhn, Werner. "Semantic engineering." *Research trends in geographic information science*. Springer, Berlin, Heidelberg, 2009. 63-76.

Ma, Xiaogang. "Knowledge graph construction and application in geosciences: A review." (2021).

Maciel, Rita Suzana P. *et al*. "Full interoperability: Challenges and opportunities for future information systems." *Sociedade Brasileira de Computação* (2017).

Mungall, Christopher J. *et al*. "k-BOOM: A Bayesian approach to ontology structure inference, with applications in disease ontology construction." *bioRxiv* (2016): 048843.

Niang, Cheikh, Béatrice Bouchou, and Moussa Lo. "Towards tailored domain ontologies." *OM*. 2010.

Noy, Natalya F., and Deborah L. McGuinness. "Ontology development 101: A guide to creating your first ontology." (2001).

Panasiuk, Oleksandra *et al*. "Verification and validation of semantic annotations." *International Andrei Ershov Memorial Conference on Perspectives of System Informatics*. Springer, Cham, 2019.

Presutti, Valentina *et al*. "eXtreme design with content ontology design patterns." *Proc. Workshop on Ontology Patterns*. 2009.

Paulheim, Heiko. "Knowledge graph refinement: A survey of approaches and evaluation methods." *Semantic web* 8.3 (2017): 489-508.

Reynolds , D.(ed.), The organization ontology, World Wide Web Consortium (W3C), 2014. https://www.w3.org/TR/vocab-org/.

Roman, Dumitru *et al*. "The euBusinessGraph ontology: A lightweight ontology for harmonizing basic company information." *Semantic Web* Preprint (2021): 1-28.

Shimizu, Cogan, Pascal Hitzler, and Clare Paul. "Ontology Design Patterns for Winston's Taxonomy Of Part-Whole Relations." *Emerging Topics in Semantic Technologies*. IOS Press, 2018. 119-129.

Shimizu, Cogan, Karl Hammar, and Pascal Hitzler. "Modular graphical ontology engineering evaluated." *European Semantic Web Conference*. Springer, Cham, 2020.

Smith, Barry et al. "The OBO Foundry: coordinated evolution of ontologies to support biomedical data integration." *Nature biotechnology* 25.11 (2007): 1251-1255.

Taylor, Kerry et al. "The semantic sensor network ontology, revamped." *JT@ ISWC*. 2019.

Wei, J. "Scalability of an Ontology-Based Data Processing System." (2018).

Wilkinson, Mark D. *et al*. "The FAIR Guiding Principles for scientific data management and stewardship." *Scientific data* 3.1 (2016): 1-9.

Wong, W., Y., 2009. “Learning Lightweight Ontologies from Text across Different Domains using the Web as Background Knowledge,” PhD thesis, University of Western Australia, School of Computer Science and Software Engineering.

A Knowledge Graph Data Model and Query Language

Kenneth Baclawski

Northeastern University

Abstract

An increasing amount of data is now available on public and private sources. Furthermore, the types, formats and number of sources of data are also increasing. The data sources have many different levels and types of structuring. Techniques for extracting, processing and analyzing such data have been developed in the last few years for managing this bewildering variety based on a structure called a knowledge graph. In this article, a new knowledge graph data model featuring formal reification is introduced and specified mathematically. This data model has many advantages compared with existing data models that have been used for representing graph structures. One important advantage is that all graph edges are reified, which can reduce the cost and complexity of storage and retrieval for knowledge graphs that have rich semantics, such as provenance, units of measure, and uncertainty specifications. In spite of the added capabilities of this knowledge graph data model, one can efficiently store knowledge graphs in existing triple stores, and existing tools can be used with only minor modifications. This article also introduces a new data language, called KGSQL, which is specifically designed for the new knowledge graph data model. Both the syntax and the denotational semantics of KGSQL are specified formally.

Introduction

THE NOTION OF A KNOWLEDGE GRAPH (KG) has emerged in the last few years to be an important semantic technology and research area. As structured representations of semantic knowledge that are stored in a graph, KGs are lightweight versions of semantic networks that scale to massive datasets such as the entire World Wide Web. Industry has devoted a great deal of effort to the development of knowledge graphs, and they are now critical to the functions of intelligent virtual assistants such as Siri and Alexa. Some of the research communities where KGs are relevant are Ontologies, Big Data, Linked Data, Open Knowledge Network, Artificial Intelligence, Deep Learning, and many others.

A KG is defined to be a labeled multigraph that has the following mathematical definition (Baclawski et al., 2020):

A node- and edge-labeled multigraph is an 8-tuple $(V, E, s, t, \Sigma_V, \Sigma_E, \ell_V, \ell_E)$ such that

1. V is a set of nodes, and E is a set of edges.
2. The functions $s: E \to V$ and $t: E \to V$ specify the source and target nodes of the edges.
3. Σ_V is a set of node labels, and Σ_E is a set of edge labels.
4. The functions $\ell_V: V \to \Sigma_V$ and $\ell_E: E \to \Sigma_E$ specify the labels of the nodes and edges.

In the definition above V and E need not be disjoint. This led to my proposal that it would be advantageous if E were a subset of V (Baclawski, 2021). In other words, every edge of the graph should itself be reified as a node of the graph. This article elaborates my proposal, explaining the rationale for introducing a new data model and a new data language; and formally specifying them.

We begin in Section 2 with the terminology for knowledge graphs that will be used. The many advantages of reifying the edges of a graph as nodes are presented in Section 3. There are other data models that have been adapted for KGs as well as many data languages. There are too many to be adequately covered in this article, but only a few are directly relevant to the aims of the work developed in this article. These data models and languages are reviewed briefly in Section 4. Our new data model is called the *knowledge graph data model* (or more briefly, the *KG model*). It formally reifies all edges, so that it has the advantages described in Section 3. We also introduce a data language that is designed for the new KG model. Our new data language is called the *knowledge graph system query language* (KGSQL). Both our new data model and our new data language are presented in Section 5. The technical details of the formal specifications of the KG model and KGSQL are given in three appendices:

1. The denotational semantics of KGSQL is specified in Appendix A. Denotational (or compositional) semantics is useful for specifying how to program an implementation of a language such as KGSQL.
2. The KG model is specified in the Distributed Ontology, Modeling, and Specification Language^{TM} in Appendix B. The specification of the KG model uses the notion of an institution that was introduced by Goguen and Burstall as a means of systematizing and relating different logical systems (Goguen and Burstall, 1983). Institutions

depend on the mathematical notions of category and functor so a brief introduction to categories and functors is also presented.
3. The formal grammar of KGSQL is in Appendix C. A parser and compiler for KGSQL queries was implemented in Java to show that the KGSQL language is consistent. The parser and compiler are available on request from the author.

The article ends with a Conclusion and Acknowledgments.

Knowledge Graph Terminology

The most common terminology for KGs is taken from the Resource Description Framework (RDF). This framework was developed to represent metadata on the Web; however, it is now being used for representing information of any kind. DF is sometimes regarded as being a specific XML representation for data; but RDF is, in fact, a data model for graph data. There are many representation languages for RDF data, all of which use the same RDF data model, most of which do not use XML. For example JSON-LD uses JSON to represent KGs.

As its name suggests, RDF is used to represent information about resources. RDF can be used to specify properties of resources and relationships between resources. A *resource* is specified using either an IRI or a blank node. An IRI is a universally unique identifier of a specific resource. A blank node specifies that a resource exists without explicitly naming it. A property of a resource is specified with a literal which can be typed or can be tagged as being in a particular natural language. In RDF, properties and relationships are specified with *statements*. Each statement specifies a subject, a predicate, and an object. Because every statement consists of three components, statements are also called *triples*. The subject and predicate are resources, and the object can be either a resource or a literal. We will write RDF statements by using angle brackets. Note that RDF uses the same term for properties and relationships; both are called "properties."

The Web Ontology Language (OWL) is a family of languages for representing ontologies. OWL is built on RDF and adds many new features and distinctions to the notions in RDF. For example OWL distinguishes relationships from properties. The former are called ObjectProperties and the latter are called DatatypeProperties.

An important relationship between resources is the one that specifies the class of a resource. For example to specify that George is a person, one might use the statement <:George rdf:type :Person>. The colons are used to specify the prefixes of resources. IRIs can be very long, so it is useful to have a mechanism for declaring abbreviations. Note that a prefix can be the empty string. Because the type relationship is so common, we will use a simpler notation for it; namely, [:George :Person].

It is often useful to focus on a subset of a graph, especially when the graph is very large. The mathematical notion for this is called a subgraph. More precisely, a *subgraph* of a graph *G* is another graph *H* formed from a subset of the vertices and edges of *G*. The vertices of *H* must include all endpoints of the edges of *H*, but may also include additional vertices of *G*.

An *edge-induced subgraph* of a graph *G* is a subgraph that does not have any additional vertices, *i.e.*, it only has vertices that are endpoints of at least one edge. The RDF standard has a notion of a *named graph* that implements an edge-induced subgraph. The name of a named graph is a blank node or IRI that can be used in RDF statements. Within an RDF database, there can be any number of named graphs. Each RDF statement in an RDF database must specify the named graph to which the RDF statement belongs. Consequently, an RDF statement is stored using a 4-tuple or quad. In spite of this, an RDF database is usually called a "triple store."

Advantages of Reifying Statements

While RDF can be used to represent KGs, it is not a perfect match. This section presents a number of examples for which a data model where all statements are reified would have significant advantages.

Higher Order Relations

One of the controversial issues of RDF is that it only has native support for binary relationships. This issue has been addressed in more recent frameworks, and these are discussed in Section 4. In RDF relationships with higher arity (*e.g.*, records) must be synthesized using properties and binary relationships. For example suppose that one wishes to implement the classic Suppliers and Parts relational database using RDF ("Suppliers and Parts", 2021). This database has three tables: Supplier, Part, and Shipment. Each record in the Supplier and Part tables is implemented by assigning a unique IRI to each supplier and part, and by asserting an RDF statement for each non-primary key attribute. The only table that is problematic is the Shipment table. This table has three columns: one each for the supplier and part, and one for the quantity of the shipment. An RDF property cannot be used to represent this table because of the third column. For example suppose that the supplier IRI is :supplier8, the part IRI is :part25, the quantity shipped is 300, and the RDF property relating them is :shipment. Then the RDF statement <:supplier8 :shipment :part25> asserts that the supplier ships the part, but does not specify the quantity shipped. To specify the quantity shipped as well as any other attributes of the shipment, one must reify the RDF statement in some way. The standard technique for RDF reification is to assert the following statements:

```
<b rdf:type rdfs:Statement>
<b rdfs:subject :supplier8>
<b rdfs:predicate :shipment>
<b rdfs:object :part25>
```

where b is either a blank node or an IRI that is unique for the reified RDF statement. The quantity may now be specified using the RDF statement <b :quantity "300"^^xsd:int>.

While this solves the problem of higher arity relations, those who have used languages that natively support higher relationship arities find the RDF limitation to binary relationships to be awkward for a number of reasons. Aside from the inefficiency of expanding a single statement into four statements, queries are now more complicated, especially if one must deal with many relationships that have been reified.

Another complication of reifying an RDF statement is that there is no connection between a statement and a reification of the statement, as the two are independent. If some statements of a relationship have been reified while others have not been reified, then queries, updates and inference rules will be much more complicated. This significantly undermines one of the supposed advantages of the RDF data model; namely, the claimed ease with which one can introduce new properties and relations. By comparison, it is relatively easy to add new columns to a relational database table without any need to update any existing queries or updates.

Yet another problem with the reification of an RDF statement is that the domain and range of a property do not apply to the reification of statements whose rdf:predicate is that property. One can specify domain and range axioms for a reification with OWL, but not with RDF, and the OWL axioms are relatively complicated.

Literals

Another example of an issue with RDF is that literals are not resources, so one cannot specify properties of literals. For example ["Hello, world!" rdfs:Literal] is necessarily true, but one cannot assert this fact using an RDF statement. It is also not possible to specify the datatype of a literal using a statement, so a separate syntax is employed to specify the datatype of a literal. For example "15"^^xsd:int specifies the number fifteen. Similarly, the (human) language of a text string is specified with a separate syntax. For example "hello"@en specifies that the language of the string "hello" is English.

One way to specify properties of literals, other than the datatype or language, is to reify the literal. A reification of a literal is called a compound literal, and its type is rdfs:CompoundLiteral. The properties of a compound literal can include its datatype and language as well as other properties such as directionality (*i.e.*, left-to-right or right-to-left), unit of measurement, *etc.* The value of such a reified literal is specified by the rdf:value property. While this solves the problem of specifying properties of literals with RDF, there are now two different ways to specify a property value: directly with the literal or indirectly with the reification of the literal. The disadvantages of this situation are then similar to the ones already mentioned in Section 3.1.

Incidentally, RDF has a non-standard extension that allows literals to be used in any slot of a statement, including the subject and predicate slots. However, this does not resolve the problem of adding properties to literals. The same string might be in many languages, might have various measurement units dependent on context, and so on.

Incremental Development

It is a common practice to develop a complex information system by using an iterative process to provide incremental improvements (Darrin and Devereux, 2017). KGs can be very complex, and there are significant practical reasons for developing them incrementally, starting from a KG represented in RDF with a minimal schema or no schema at all, and later gradually improving the schema (Berg-Cross, 2021). However, one problem with such an approach is that improving the KG with an improved schema will often require a wholesale restructuring of the KG to a new version that is not backwardly compatible with earlier versions. Sections 3.1 and 3.2 give examples in which improving the schema requires changing the structure of the KG. For example, the initial KG might have statements with properties such as distances, speeds, and weights, expressed as raw numbers. Adding appropriate units to these statements requires designing a schema that allows for specifying the unit of measure for such statements, as well as other properties such as uncertainty and provenance. Reifying all statements allows one to incrementally add such properties to statements without any restructuring. While the KG model cannot entirely eliminate the need for KG restructuring during an incremental development process, it can make it much easier in many important cases.

Another advantage of the KG model is that it reduces the number of possible designs for incrementally adding properties to statements and collections of statements. While one can easily design an RDF schema that will allow one to add units of measure to a statement, there are several ways to do this, as noted in Section 3.2. This can hamper interoperability as well as incremental development efforts that wish to incorporate existing KGs (Berg-Cross, 2021). Since the KG model reifies edges, there are fewer designs for adding properties to statements, and they are more easily accommodated during incremental development and interoperation. For example, two KGs might use different properties for specifying a unit of measure, but the basic structure is the same. Again, the KG model does not entirely solve this problem, but it could make it easier to solve.

Graph Languages with Edge Quoting and Reification

While there are many graph query languages, only a few have mechanisms for simplifying edge reification. In this section we review these languages.

The Property Graph Query Language (PGQL) is a graph query language built on top of SQL. The purpose PGQL is to extend SQL to have graph pattern matching capabilities (PGQL, 2021). A *property graph* is a graph in the usual sense of nodes and edges, such that nodes and edges can have properties. A PGQL schema is similar to an SQL schema except that tables are either vertex tables or edge tables. Each edge table has a source table and a destination table. The difference between the PGQL and KG models is that in the KG model all statements, including properties, are reified. Another difference is that the type of a KG resource is specified with a statement, while in PGQL the type of a resource is the table it belongs to. So a KG resource can have many types, and a resource can change its type. Changing the table of an entity is not meaningful in SQL and PGQL.

The RDF-star is an unofficial draft data model that is intended to simplify specifying RDF reification (RDF-star, 2021). The corresponding data language is SPARQL-star. The feature that RDF-star adds to RDF is the ability to *quote* a triple. Quoted triples are specified in double angle brackets. Quoting a triple is not the same as the reification of the triple. If the same quoted triple appears multiple times, then all of them are necessarily the same. By contrast, the same triple can be reified more than once in the KG model, and each reification will have its own triple identifier. To associate an identifier with a quoted triple, one must specify the identifier with another triple. One can then use this identifier in other triples. While this does reify the triple, there is now a distinction between the quoted triple and its reification. Moreover, the property that is used to reify the quoted triple is not unique. Indeed, within the same RDF-star store one could reify the same triple using more than one property.

One can nest RDF-star triples to any depth. All of the triples in such an expression are quoted except for the outermost triple. One cannot have infinite nesting of RDF-star triples, so that it appears to be impossible to specify a circular structure. However, if one reifies the quoted RDF-star triples, then one can define circular structures.

The Knowledge Graph Data Model and System Query Language

A common feature of the issues with RDF discussed in Section 3 is the use of reification to resolve the issue. This suggests that a language where reification is provided in a seamless manner would have many advantages. While some languages have attempted to deal with this issue to some degree, as discussed in Section 4, it might be worthwhile to consider a more dramatic approach; namely, reify *every* statement and introduce a data language that leverages the reifications. We call this language KGSQL, and specify both the KG model and KGSQL in this section along with some applications.

The Knowledge Graph Data Model

We define the KG model by specifying a concrete realization of the notion of a knowledge graph as defined in Section 1. This realization uses ordinary set theory. The category theoretic specification is given in Appendix B.

We start by specifying the different kinds (or, more precisely, sorts) of the nodes that can be in a knowledge graph as follows:

- *GId* is the set of all possible global resource identifiers;
- *LId* is the set of all possible local resource identifiers;
- *Lit* is the set of literal strings;
- *Num* is the set of double-precision numbers;
- *Bool* is the set of Boolean values;
- $\perp$ denotes the undefined result; and
- *V* is the set of all possible variables.

It is assumed that the sets *GId, LId, Lit, Num, Bool,* and *V* are disjoint and do not contain $\perp$, and that *V* is infinite. The union $GId \cup LId$ is the set of resource identifiers and will be written *Id*. The union $Id \cup Lit \cup Num \cup Bool$ will be written *Res*.

A *knowledge graph* is a relation $G \subseteq Res \times Id \times Res \times Id$ such that if $(s,p,o,e), (s',p',o',e) \in G$ then $s=s'$, $p=p'$ and $o=o'$. In other words the fourth component is a unique column of the relation. The elements of a KG are called *statements*, and the components are called the *subject*, *predicate*, *object*, and *statement identifier*, respectively.

The set of all elements of *Res* that occur as one or more of the components of an edge of a knowledge graph *G* will be written *G.node*. The type of a node that is not a statement identifier is specified by a statement whose property is rdf:type. The type of a statement is its property (*i.e.*, its edge label), and it is not necessary to have an explicit edge in *G* that specifies this fact. A node can have more than one type. Table 1 shows the correspondence between the terms of a multigraph and the terms of a KG. In this table, *FinSet(S)* is the collection of all finite subsets of a set *S*.

Multigraph	Knowledge Graph
V	*G.node*
E	*G*
s	The subject of a statement
t	The object of a statement
Σ_V	*FinSet(G.node)*
Σ_E	*FinSet(G.node)*
ℓ_V	The types or properties of a node
ℓ_E	The properties of a statement

Table 1: Relationship between multigraph and knowledge graph terminology

The realization of the notion of a KG not only defines the KG model it also gives an example of an implementation; namely, one can implement a KG with a single relational table with four columns. The fourth column is the statement identifier. Another implementation is to store a KG in a triple store. This is possible because, in practice, triple stores actually store quads. The fourth component is the name of the named graph. The fourth component could store both the name of the named graph and the statement identifier by concatenating the name of the named graph with a unique identifier for the statement within the named graph, separating the concatenated strings with a special character. In other words the statement identifier includes the name of the named graph to which it belongs. Of course this strategy requires that the statement identifiers have a particular form.

The Query and Update Languages

We now discuss the syntax of KGSQL. The syntax was designed to be as similar as possible to SPARQL; indeed, most SPARQL commands should also be KGSQL commands. The most important additional feature is the ability to explicitly reference the identifier of any statement. KGSQL supports SELECT, ASK, CONSTRUCT, INSERT, and DELETE commands. The SELECT command finds all collections of edges in the KG that satisfy the WHERE clause and returns the selected variables. The ASK command is the same as the SELECT command except that it only returns whether there were any matching edges. The ASK command can be used as a subquery in the where clause of another command. The CONSTRUCT command is the same as the SELECT command except that the selected variables are used to specify a set of edges in a graph which are then returned. The INSERT command is the same as the CONSTRUCT command except that the constructed edges are stored in the KG database. The INSERT command can modify edges in the KG database so that the INSERT command is also an update command. The DELETE command is the same as the SELECT command except that the variables that are returned are the statement identifiers of the edges that are removed from the KG database.

The WHERE clause of any command consists of patterns and filters. A pattern specifies a subject, predicate and object, which can be variables or constants. A variable is indicated with an initial question mark. CONSTRUCT and INSERT commands use patterns to specify the edges to be constructed or inserted.

As in SPARQL, some abbreviations are supported. If several patterns have a common subject, then one can specify the subject followed by a sequence of predicate-object pairs separated by semicolons. If several patterns have the same subject and predicate, then they can be followed by a sequence of objects separated by commas. Unlike SPARQL, the subject and object can be bracketed expressions which specify an instance and its class, either or both of which can be a variable. The class can be a pipe-delimited sequence of classes, denoting a union of the classes. A union does not allow variables. A bracketed expression can specify a constant or variable in between the instance and the class (or class union), which represents the statement identifier of the type statement. The predicate can also be a bracketed expression, but it cannot specify a third slot in the middle. The bracketed expression of a predicate specifies the statement identifier and the property. In other words it is as if properties are classes whose instances are statements.

Patterns are a textual notation for specifying KGs. Another way to specify a KG is with a drawing of the nodes and edges. Since the edges are directed from the subject to the object, one uses arrows for edges. The property of an edge is shown next to the edge. Since edges are also nodes, it is possible for an edge to start at another (or the same) edge and also to end at an edge. For example the Suppliers and Parts example in Section 3.1 could be specified in the KG model as follows:

:supplier14 [?e :shipment] :part34 .
?e :quantity ["500" xsd:decimal] .

and it could be drawn using the diagram in Figure 1.

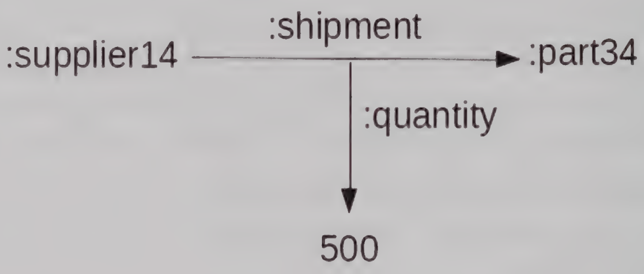


Figure 1: The Suppliers and Parts Example

The Suppliers and Parts database is unrealistic. In practice one shipment may include several parts but would always be to one customer on one date, organized as an invoice. So it would be better to make the customer the main attribute of a shipment as in Figure 2.

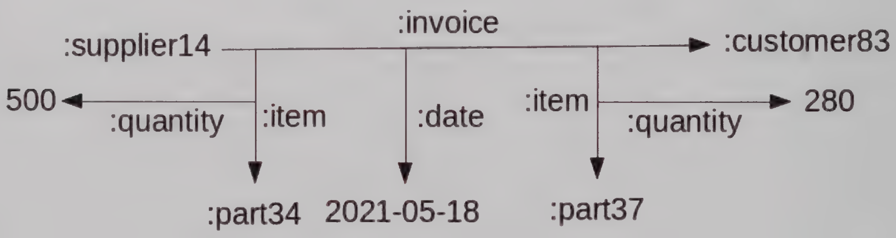


Figure 2: Example of an invoice

All the edges in Figure 2 are reified so one can easily add additional information about each one as needed. For example one could add context information such as provenance and uncertainty (Baclawski *et al.*, 2018). As another example, one could add rationales which could be used for the purposes of explanation (Baclawski *et al.*, 2019; Baclawski, 2020).

The reason for not allowing a middle slot in a bracketed expression for the predicate is subtle. In the KG model, each statement is an instance of its predicate. However, if the relationship between each statement and its predicate were always reified, then this would entail an infinite sequence of reifications. For example suppose that we have the statement :supplier8 :name "QWPNW". Let :e1 be its statement identifier. The predicate of the statement identified by :e1 is :name. So :e1 has the type :name. If this fact is reified, then the statement :e1 rdf:type :name would have a statement identifier. Suppose that :e2 is the statement identifier of :e1 rdf:type :name. Then the predicate of :e2 is rdf:type. If this fact is reified, then :e2 rdf:type rdf:type would also have a statement identifier, and so on, as drawn in Figure 3. The result is an infinite sequence of distinct statement identifiers. Presumably, one could find a way to manage such infinite sequences, since they all have the same simple form, but it is simpler to agree to omit them.

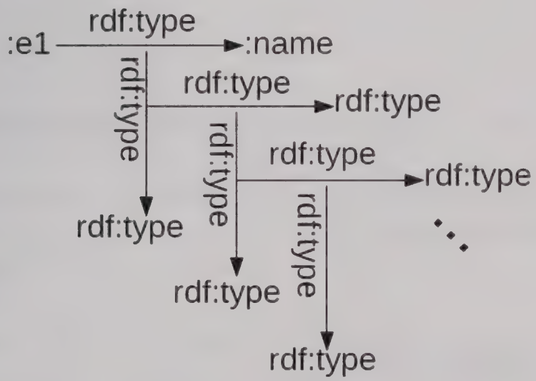


Figure 3: An infinite sequence of reifications

A pattern can specify a multiplicity for following paths in a KG. A multiplicity is one or two integers or asterisks, representing the minimum and maximum number of edges to follow. An asterisk represents an unlimited number of edges. If there is one integer or asterisk, then the minimum and maximum are the same. The multiplicity can be in one of two places in a pattern. If the multiplicity is after the second slot, then the multiplicity specifies that one should follow statement identifiers; and if the multiplicity is after the third slot, then the multiplicity specifies following objects.For example in the drawing in Figure 4, the pattern

:a prop ?x {6}

would set ?x to :b, and the pattern

:c prop {6} ?x

would set ?x to :d. This notation is analogous to the notation for array selection in programming languages where it would look something like this: x = c.prop[6].

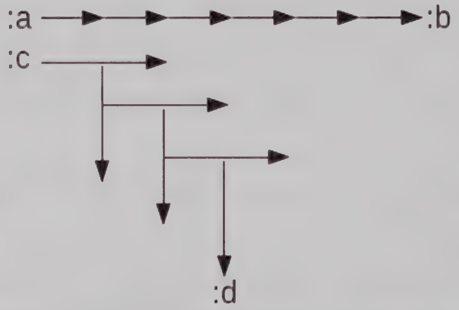


Figure 4: Path following in a knowledge graph

A negative multiplicity goes in reverse (*i.e.*, via the inverse relationship). For example, the pattern

?x prop :b {-6}

would set ?x to :a, and the pattern

?x prop {-6} :d

would set ? x to :c.

If the maximum or minimum is an asterisk (*i.e.*, unbounded), then the result is a transitive closure.

The formal syntax of the initial version of KGSQL is in Appendix C. Only the grammar is shown in Appendix C. The lexical rules were omitted for simplicity. The full grammar is available at kgsql.org/KGSQL.g4. The syntax notation is that of Antlr (Parr, 2014). Not all features of SPARQL were included in the initial version KGSQL, since the initial version is only a proof of concept. Other features of SPARQL will be added in later versions.

Sequences

One of the advantages of reifying all statements is that one can construct sequences (also called linked lists) much more easily and efficiently. Moreover, one can specify sequences with any property as the sequence property. In RDF, a sequence is specified using the built-in properties rdf:first and rdf:rest, and the built-in resource rdf:nil. For example the sequence (:Kim :Greer :Qing) is specified with the following statements:

```
<a rdf:first :Kim>
<a rdf:rest b>
<b rdf:first :Greer>
<b rdf:rest c>
<c rdf:first :Qing>
<c rdf:rest rdf:nil>
```

In KGSQL, a sequence is specified like this:

```
c [a :prop] :Kim .
a [b :prop] :Greer .
b [c :prop] :Qing .
```

where :prop could be any property. Note that this sequence is circular and that no notion of nil is necessary for nonempty sequences. Of course one does require a way to specify an empty sequence, such as the built-in resource kgsql:empty. In KGSQL, the notation for this sequence is [(:Kim :Greer :Qing) :prop]. The drawing for this sequence is shown in Figure 4.

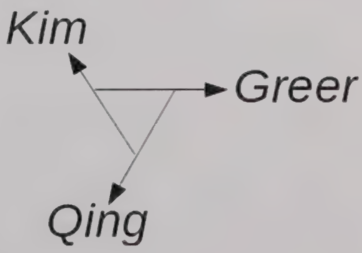


Figure 4: Example of a three-element sequence

Comparison with SQL

In this section, we discuss how KGSQL differs from SQL, or more precisely comparing the KG model with the relational model. Many comparisons of triple stores and the relational databases have been published, such as (Comparison, 2013). These comparisons generally focus on pragmatic issues of the use of different products rather than the conceptual differences between the models. While a pragmatic focus is valuable, it might also be worthwhile to consider the conceptual differences, and we do so in the following.

One major conceptual difference between KGs and the relational model is the way that entities are conceptualized. In a KG, the entities are the nodes in the graph. In RDF entities are called resources and are specified with IRIs. By contrast, the entities in the relational model are records (also called rows or tuples). A record is specified with its primary key. To see why this distinction is more than just a question of data representation, consider how queries for the two models are mapped to the variables of predicate logic. For the relational model, the variables represent records. In other words, when one quantifies in the relational model (using either existential or universal quantification), the quantification is over the records of a specific relation (or table). This is fundamental to any relational query language. To borrow a term from programming languages, relational variables are strongly typed. Indeed, in SQL, by default, the name of a variable is the name of a table; non-default variable names are necessary only if a query is quantifying over the same table more than once.

By contrast the variables of a KG query vary, in principle, over all possible nodes of a graph. To the extent that a variable is typed at all, the type constraint is just one more statement which is no more fundamental than any other constraint. In particular a type constraint need not be imposed at all, and if it is, then it is imposed explicitly. The relational model has no such flexibility.

Some relational database systems maintain a rowid (also called a "rid") that uniquely identifies each record in a table. This appears to be analogous to a statement identifier. However, while a rowid can be accessed in a query, it is set and maintained by the database system. More importantly, the database can change a rowid when a record is updated and can reuse the rowid of a deleted record.

Another way to look at the distinction between the KG and relational models is to treat a KG as being a single relation whose records are the nodes and whose primary key is the node identifier. This single relation has infinitely many multi-valued columns, one for every possible property or relationship. This relation is not quite a relation in the classical sense because it has multi-valued columns, but many relational databases support such columns. In any case, the example is only a conceptual one as it is not at all practical, and it completely omits the reification of statements that is fundamental to KGSQL.

A more practical and complete implementation of a KG with a relational database is to use a four-column table to represent the statements, with an additional column to specify either a named graph identifier (for the RDF data model) or a statement identifier (for the KG model). Let T(G) be the table for a knowledge graph G. This implementation is more promising than the previous one, and some products use this technique. However, it would be misleading to view a KG as being T(G) or even being analogous to T(G) as is commonly mentioned, such as in the Wikipedia article (SPARQL, 2021). The difficulty with this analogy is that from a relational point of view the entities of T(G) are the *records* of T(G), not the entries in any of the columns of one of the records. This confusion is especially significant for the RDF data model because RDF statements, which correspond to the records of T(G), are never entities of the RDF data model. The distinction is not merely conceptual as it affects how one programs queries for the RDF model compared with relational queries. In the relational model one is iterating over records in tables while in the RDF model one is iterating over resources. It is somewhat less significant for the KG model because the records of T(G) are always KG nodes and hence are entities, but there will also be many other entities, so programming a KGSQL query will also differ from programming an SQL query.

In summary the KG and relational models for data differ not only pragmatically but also conceptually. The two models have different notions of what an entity is in the model. As a result, the two models tend to have different approaches to design, development, and employment. While the RDF data model and the KG model are similar, the differences between them are also likely to result in different approaches to design, development and employment.

Conclusion and Future Work

We have developed a new data model specifically designed for knowledge graphs. In this regard the development of the KG model is analogous to the development of the relational model by Codd and many others, which was designed for relational (tabular) data (Codd, 1970). While the KG model is designed for KGs, it is effective for representing data of other kinds, such as hierarchical and tabular data. We have also developed KGSQL, a data language designed for the KG model. KGSQL helps to resolve some of the disadvantages of existing data languages for KGs. Both the KG model and the KGSQL language are formally specified, with the details given in the appendices.

The KG model and KGSQL are concerned only with data, not the schema. This has the advantage that the data and schema are separate concerns. In principle a KG could have more than one schema; for example, different schemas might be used by different communities for their own purposes. That said, an interesting future direction for the work on the KG model would be to develop a schema language for specifying semantics. Ideally, the schema language would also use the KG model, so that the schema could be written in KGSQL.

One of the most powerful insights of the KG model is that properties (including relationships) are classes whose instances are statements. In OWL, properties and classes are disjoint. In the KG model not only are properties and classes not disjoint, but the properties are a subset of the classes. This idea makes it possible for properties to be domains and ranges of other properties, thereby allowing more opportunities for specifying the semantics of data expressed as KGs. An interesting future project would be to develop this idea both practically and formally.

Acknowledgments

The author would like to thank Gary Berg-Cross, Ravi Sharma, Janet Singer, Ram D. Sriram, and Andrea Westerinen for many helpful comments that improved this article.

References

K. Baclawski (2020). Decision Rationales as Models for Explanations. In *J. Wash. Acad. Sci.*106(4):107-124.

K. Baclawski (2021). Introduction to KGSQL: A Knowledge Graph System Query Language, July 7 2021. Retrieved 1 October 2021 from https://bit.ly/3xocWEj.

K. Baclawski, M. Bennett, G. Berg-Cross, C. Casanave, D. Fritzsche, J. Ring, T. Schneider, R. Sharma, J. Singer, J. Sowa, R.D. Sriram, A. Westerinen and D. Whitten (2018). Ontology Summit 2018 Communiqué: Contexts in Context. In *Journal of Applied Ontology*13(3):181-200. IOS Press, The Netherlands. (July, 2018)

K. Baclawski, M. Bennett, G. Berg-Cross, D. Fritzsche, R. Sharma, J. Singer, J. Sowa, R.D. Sriram, M. Underwood and D. Whitten (2019). Ontology Summit 2019 Communiqué: Explanation. In *Applied Ontology*. IOS Press, The Netherlands. DOI: 10.3233/AO-200226.

K. Baclawski, M. Bennett, G. Berg-Cross, D. Fritzsche, R. Sharma, J. Singer, J. Sowa, R.D. Sriram, M. Underwood, and D. Whitten (2020). Ontology Summit 2020 Communiqué: Knowledge Graphs. *Applied Ontology*, 16(2):229-247, April 2020.

Comparison of triple stores vs relational databases (2013). Retrieved 2 October 2021 from https://bit.ly/3ooqIoV.

G. Berg-Cross (2021). Introduction to Harmonizing Definitions and the EnvO Ontology. Retrieved 1 June 2021 from https://bit.ly/3pnkwdG.

E. Codd (1970). A Relational Model of Data for Large Shared Data Banks. In *Communications of the ACM* 13 (6) 377-387.

M. Darrin and W. Devereux (2017). The agile manifesto, design thinking and systems engineering. In *Annual IEEE International Systems Conference (SysCon)*, pages 1-5. IEEE.

J. Goguen and R. Burstall (1983). Introducing institutions. In *Proc. Carnegie Mellon Workshop on Logic of Programs*, volume 164, pages 221-256.

D. Jurafsky and H.J. Martin (2000). Speech and language processing: an introduction to natural language processing, computational linguistics, and speech recognition. Upper Saddle River, N.J.: Prentice Hall. ISBN 978-0-13-095069-7.

T. Parr (2014). Antlr website. Retrieved 2 October 2014 from https://www.antlr.org.

The Property Graph Query Language (PGQL) website (2021). Retrieved 2 October 2021 from https://pgql-lang.org/.

RDF-star and SPARQL-star (2021). Draft Community Group Report 01 October 2021. Retrieved 2 October 2021 from https://bit.ly/3F6ORX3.

Suppliers and Parts (2021). Wikipedia article. Retrieved 2 October 2021 from https://bit.ly/3B4wv6p.

Wikipedia article on SPARQL (2021). Retrieved 2 October 2021 from https://bit.ly/3B4eXHC.

Appendix A Denotational Semantics

A KGSQL query is a collection of patterns (or more precisely, pattern instances) and a collection of filters that constrain the results of a query. A pattern is a statement where each of the components may be either a constant or a variable. A constant is a resource identifier, string, number or Boolean value. A result of a query is a function from a finite set of variables to a constant. The result set of a query is a multiset of results. We now specify using denotational (compositional) semantics how to determine the result set of a query.

We begin with some mathematical notation in A.1. In A.2, the evaluation of filter expressions is specified recursively, and in A.3, the semantics of a query is defined.

A.1 Mathematical Background

In this section we will use the notation for KGs introduced in Section 5.1. In particular, we use the set *Res* of all possible nodes that can be in a KG.

A *result* (of a query) is a function $f: W \rightarrow Res$, where $W \subset V$ is a finite set of variables. The domain of a result f is written $dom(f)$. The domain of a result can be empty, and there is a unique result with an empty domain (*i.e.*, the empty function). The *projection* of a result $f: W \rightarrow Res$ to a subset $U \subseteq W$ is the restriction of f to U and is written $\pi_U f$. Two results $f: W \rightarrow Res$ and $g: Y \rightarrow Res$ are said to be *compatible* if they coincide on the intersection of their domains, *i.e.*, $\forall v \in W \cap Y, f(v) = g(v)$. The *join* of compatible results f and g is the unique function $h = f \bowtie g: W \cup Y \rightarrow Res$ such that $\pi_W h = f$ and $\pi_Y h = g$.

A *multiset* is a set that can have repeated elements. A *result set S* is a multiset of results all of which share the same domain which is written *dom(S)*. The projection of a result set with domain *W* to a subset $U \subseteq W$ is the multiset $\pi_U(S) = \{\pi_U f: U \rightarrow Res | f \in S\}$ such that if several results become the same after projection to *U*, then their multiplicities are summed. The *join* of two result sets *S* and *T* is the multiset $S \bowtie T = \{f \bowtie g | f \in S \text{and} g \in T \text{are compatible}\}$. The join of results sets accounts for results that occur more than once. This means that if $f \in S$ has multiplicity *m* and if $g \in T$ has multiplicity *n,* then the multiplicity of $f \bowtie g \text{in} S \bowtie T$ is the product *mn*.

A *pattern* (or more precisely, a *pattern* instance) is a quadruple *P = (s,p.o,e)* such that the four components are in $V \cup Res$, *i.e.*, $s, p, o, e \in Res \cup V$. The set of variables that occur in a pattern *P* will be written *var(P).* We extend a result set $f: W \rightarrow Res$ to all of $V \cup Res$ by setting *f(x)* to *x* for every $x \notin dom(f)$. Similarly, we define *f(P)* for a pattern $P = (s, p, o, e)$ to be $(f(s), f(p), f(o), f(e))$. In other words, for each component of a pattern, if the component is in *dom(f)* then apply *f* to the component; otherwise, leave the component alone.

A pattern is the primary KGSQL query mechanism. Given a knowledge graph *G* and pattern *P,* the semantics of *P* with respect to *G* is the result set

$$G(P) = \{f: var(P) \rightarrow Res | f(P) \in G\}$$

The syntax for a pattern $P = (s, p, o, e)$ in a KGSQL query is

s [e p] o

We will use the following notation for the semantics of a primary pattern in a *KGSQL* query:

$$[\![\text{s [e p] o}]\!]_G = G(s, p, o, e).$$

A.2 Query Filters

A *filter* is a Boolean expression that limits the result set of a query to contain only the results that satisfy the filter expression. The evaluation of an expression *expr* with respect to a result *f* and knowledge graph *G* is written $eval(expr, f, G)$. Filter expressions in a KGSQL query use a typical syntax, except that the logical operators (*i.e.*, AND, OR and NOT) can be specified using either infix operators or functions. The logical infix operators are left-associative short-circuit operators, whereas the logical functions are not short-circuit operators. The evaluation of expressions is defined recursively as follows.

1. $eval(expr_1 || expr_2 . f, G) =$
$$\begin{cases} \text{true}, & \text{if} eval(expr_1, f, G) = \text{true}, \\ \text{false}, & \text{if} eval(expr_1, f, G) = \text{false and } eval(expr_2, f, G) = \text{false}, \\ \perp, & \text{otherwise.} \end{cases}$$

2. $eval(expr_1 \&\& expr_2 . f, G) =$
$$\begin{cases} \text{false}, & \text{if} eval(expr_1, f, G) = \text{false}, \\ \text{true}, & \text{if} eval(expr_1, f, G) = \text{true and } eval(expr_2, f, G) = \text{true}, \\ \perp, & \text{otherwise.} \end{cases}$$

3. For a binary operator op ∈ {=,!=},
$eval(expr_1 \text{op} expr_2 . f, G)$
$$= \begin{cases} eval(expr_1, f, G) \text{op} eval(expr_2, f, G) & \text{if both evaluations are defined,} \\ \perp, & \text{otherwise.} \end{cases}$$

4. For a binary operator op ∈ {<,>,<=,>=}, $eval(expr_1 \text{op} expr_2 . f, G) =$
$$\begin{cases} eval(expr_1, f, G) \text{op} eval(expr_2, f, G) & \text{if both evaluations are in} Lit, \\ eval(expr_1, f, G) \text{op} eval(expr_2, f, G) & \text{if both evaluations are in} Num, \\ \perp, & \text{otherwise.} \end{cases}$$

5. For a binary operator op ∈ {+,-,*},
$eval(expr_1 \text{op} expr_2 . f, G)$
$$= \begin{cases} eval(expr_1, f, G) \text{op} eval(expr_2, f, G) & \text{if both evaluations are in} Num, \\ \perp, & \text{otherwise.} \end{cases}$$

6. $eval(expr_1 / expr_2 . f, G) =$

7. $eval(+expr . f, G) = eval(expr, f, G).$

8. $eval(-expr . f, G) = \begin{cases} -eval(expr, f, G) & \text{if the evaluation is in} Num, \\ \perp, & \text{otherwise.} \end{cases}$

9. $$eval(!expr.f,G) = \begin{cases} \text{false}, & \text{if} eval(expr,f,G) = \text{true}, \\ \text{true}, & \text{if} eval(expr,f,G) = \text{false}, \\ \bot, & \text{otherwise}. \end{cases}$$

10. For a function g that can take n arguments,
$eval(g(expr_1, expr_2, \ldots, expr_n), f, G) =$

11. For a variable $v \in V$, $eval(v.f,G) = \begin{cases} f(v) & \text{if } v \in dom(f), \\ \bot, & \text{otherwise}. \end{cases}$

12. For a constant $c \in Res$, $eval(c,f,G) = c$.

13. For a WHERE clause W, $eval([\![\text{ask}W]\!]_G, f, G) =$
$$\begin{cases} \text{true}, & \text{if } [\![W]\!]_G \neq \emptyset, \\ \text{false}, & \text{if } [\![W]\!]_G = \emptyset, \\ \bot, & \text{otherwise}. \end{cases}$$

A.3 Query Semantics

The semantics of a KGSQL query is specified in this section. A query has both query clauses and filter clauses. A filter clause is an expression in the variables occurring in the query clauses. The filter expressions are evaluated for each result of the query clauses, and only the results for which the filter expressions all evaluate to true are returned. During this process, if a filter expression is undefined, then the entire query is undefined.

The denotational semantics of a KGSQL query with respect to a knowledge graph G is specified recursively below. The symbols s, p, o, e, r, and c are elements of $Res \cup V$, and the symbols v and w are variables in V that do not occur among the variables in the query.

1. $[\![\text{s [e p] o}]\!]_G = G(s,p,o,e)$

2. $[\![\text{[r e c]}]\!]_G = [\![\text{r [e rdf:type] c}]\!]_G$

3. For a finite subset $\{c_1, c_2, \ldots, c_n\} \subseteq Id$, $[\![\text{r e } c_1|c_2|\ldots|c_n]]\!]_G = \bigcup_{i=1}^{n} [\![\text{r e } c_i]\!]_G$

4. For an integer m, $[\![s\ [e\ p\ ]\ o\ \{m\}]\!]_G =$
   a) $\emptyset$, if $m = 0$,
   b) $[\![s\ [e\ p]\ o]\!]_G$, if $m = 1, -1$,
   c) $[\![s\ [v_1\ p]\ w_1]\!]_G \bowtie_\bowtie [\![w_1\ [v_2\ p]\ w_2]\!]_G \bowtie \cdots \bowtie [\![w_{m-1}\ [e\ p]\ o]\!]_G$, if $m >$ 1.
   d) $[\![w_1\ [v_1\ p]\ o]\!]_G \bowtie [\![w_2\ [v_2\ p]\ w_1]\!]_G \bowtie \cdots \bowtie [\![s\ [e\ p]\ w_{-m-1}]\!]_G$, if $m < -$ 1,

5. For integers $m \leq n$, $[\![s\ [e\ p\ ]\ o\ \{m..n\}]\!]_G = \cup_{i=m}{}^n[\![s\ [e\ p\ ]\ o\ \{i\}]\!]_G$

6. For an integer m, $[\![s\ [e\ p\ ]\ o\ \{m..*\}]\!]_G = \cup_{i\geq m}[\![s\ [e\ p\ ]\ o\ \{i\}]\!]_G$

7. For an integer n, $[\![s\ [e\ p\ ]\ o\ \{*..n\}]\!]_G = \cup_{i\leq m}[\![s\ [e\ p\ ]\ o\ \{i\}]\!]_G$

8. $[\![s\ [e\ p\ ]\ o\ \{*..*\}]\!]_G = \cup_i[\![s\ [e\ p\ ]\ o\ \{i\}]\!]_G$

9. For an integer m, $[\![s\ [e\ p\ ]\ \{m\}\ o]\!]_G =$
   e) $\emptyset$, if $m = 0$,
   f) $[\![s\ [e\ p]\ o]\!]_G$, if $m = 1, -1$,
   g) $[\![s\ [v_1\ p]\ w_1]\!]_G \bowtie_\bowtie [\![v_1\ [v_2\ p]\ w_2]\!]_G \bowtie \cdots \bowtie [\![v_{m-1}\ [e\ p]\ o]\!]_G$, if $m > 1$,
   h) $[\![w_1\ [v_1\ p]\ o]\!]_G \bowtie [\![w_2\ [v_2\ p]\ v_1]\!]_G \bowtie \cdots \bowtie [\![s\ [e\ p]\ v_{-m-1}]\!]_G$, if $m < -1$,

10. For integers $m \leq n$, $[\![s\ [e\ p\ ]\ \{m..n\}\ o]\!]_G = \cup_{i=m}{}^n[\![s\ [e\ p\ ]\ \{i\}\ o]\!]_G$

11. For an integer m, $[\![s\ [e\ p\ ]\ \{m..*\}\ o]\!]_G = \cup_{i\geq m}[\![s\ [e\ p\ ]\ \{i\}\ o]\!]_G$

12. For an integer n, $[\![s\ [e\ p\ ]\ \{*..n\}\ o]\!]_G = \cup_{i\leq m}[\![s\ [e\ p\ ]\ \{i\}\ o]\!]_G$

13. $[\![s\ [e\ p\ ]\ \{*..*\}\ o]\!]_G = \cup_i[\![s\ [e\ p\ ]\ \{i\}\ o]\!]_G$

14. $[\![s\ p\ o]\!]_G = \pi_{var\{s,p,o\}}[\![s\ [v\ p]\ o]\!]_G$

15. If the subject or object or both are bracketed expressions, then the result set is obtained by join. For example, $[\![[s\ e\ c]\ p\ o]\!]_G = [\![[s\ e\ c]]\!]_G \bowtie [\![s\ p\ o]\!]_G$

16. For a set of clauses $Q = \{Q_1, Q_2, ..., Q_n\}$ as in cases (1) to (15), $[\![Q]\!]_G = [\![Q_1]\!]_G \bowtie [\![Q_2]\!]_G \bowtie \cdots \bowtie [\![Q_n]\!]_G$

17. For a set of clauses $Q = \{Q_1, Q_2, ..., Q_n\}$ and an expression *expr* in the variables *var(Q)*,

$[\![\text{Q . filter}(expr)]\!]\text{G} =$

a) $\{f \in [\![\text{Q}]\!]_G \mid eval(expr, f, G) = \text{true}\}$, if every evaluation is defined, and

b) $\perp$, if $eval(expr, f, G) = \perp$ for any $f \in [\![\text{Q}]\!]_G$

18. For a WHERE clause W consisting of a set of query clauses $\text{Q} = \{\text{Q}_1, \text{Q}_2, ..., \text{Q}_n\}$
and a sequence of filter expressions $expr_1, expr_2, ..., expr_n$, in the variables $var(\text{Q})$,
$[\![\text{W}]\!]_\text{G} = [\![\text{Q . filter}(expr_1 \ \&\& \ expr_2 \ \&\& \ ... \ \&\& \ expr_n)]\!]_\text{G}$

9. For a WHERE clause W and a nonempty set of variables $v_1, v_2, ..., v_n \in V$,
$[\![\text{select } v_1, v_2, ..., v_n \text{ where } W]\!]_\text{G} =$

a) $\pi_{\{v1, v2, ..., vn\}}[\![\text{W}]\!]_\text{G}$, if $[\![\text{W}]\!]_\text{G}$ is defined

b) $\perp$, if $[\![\text{W}]\!]_\text{G} = \perp$

Appendix B Category Theory and Institutions

A category is a labeled directed graph with an associative composition operation. In this section, the notions of category and functor are defined using computer science notation rather than the traditional notation from mathematics.

Category theory abstracts the more concrete notions of function and function composition, which we now discuss because nearly the same notation is used for category theory. For a set S, the identity function on S is denoted $\mathbf{1}_S$. For a function $f: S \to T$, the *domain* of f is S and the *codomain* of f is T. For sets S, T, U, and functions $f: S \to T$ and $g: T \to U$, the composition of f and g is a function $h: S \to U$ such that for each element x in S, $h(x) = g(f(x))$. The composition is written either $g \circ f$ or as $f;g$. The latter is generally preferred by computer scientists because it is analogous to the notation in most programming languages for the composition of successive statements in a program. If F and G are sets of functions, the set of pairs of functions of F and G that are composable will be written $F \bowtie G$. In other words, $F \bowtie G = \{(f, g) \mid f \in F, g \in G$, and the codomain of f is the same as the domain of $g\}$.

The notation for category theory differs from the notation for sets and functions in a few ways. Instead of functions, one has morphisms. Instead of the domain and codomain of a function, one speaks of the source and target of a morphism. The notation for the composition of morphisms is the same as the notation for functions, except that the formula for the composition need not hold.

A *category* **C** consists of the following components:

1. A collection **C.ob** of *objects.*
2. A collection **C.hom** of *morphisms.*
3. A function **C.src**: **C.hom** → **C.ob** that specifies the *source* of each morphism.
4. A function **C.tar**: **C.hom** → **C.ob** that specifies the *target* of each morphism.
5. A function **C.id**: **C.ob** → **C.hom** that specifies the *identity morphism* of each object.
6. A function **C.comp**: **C.hom** ⋈ **C.hom** → **C.hom** that specifies morphism composition such that composition is associative, and the composition of an identity morphism with another morphism is equal to the other morphism.

It is a common practice to specify category theory axioms and results using commutative diagrams. A diagram of objects and morphisms is said to be commutative when for every pair of objects in the diagram, the composition of every path of morphisms from the first object to the other yields the same morphism. These diagrams are a way of visualizing axioms and results using graphs. Both the nodes and the edges of a commutative diagram are labeled, so a commutative diagram is a knowledge graph. For example Figure 5 specifies that the source and target of an identity morphism are the object of the identity morphism. In Figure 5 there are three paths from the upper left corner to the lower right corner. The diagram is commutative when all three paths yield the same morphism. In other words, when $\mathbf{C.id;C.src} = \mathbf{1}_{\mathbf{C.ob}} = \mathbf{C.id;C.tar}$.

$$\begin{tikzcd}
C.ob \arrow[r, "C.id"] \arrow[d, "C.id"'] \arrow[dr, "1_{C.ob}"] & C.hom \arrow[d, "C.src"] \\
C.hom \arrow[r, "C.tar"'] & C.ob
\end{tikzcd}$$

Figure 5: Example of a commutative diagram

The collection of all sets and functions between them forms a category **Set**. If the source and target of every morphism of a category **C** are interchanged, *i.e.*, the direction of every morphism is reversed, then the result is again a category which is written $\mathbf{C}^{op}$.

While a category is a labeled directed graph, the converse is not necessarily true, since directed graphs do not generally have a composition operation. However, finite paths in a graph can be composed, and the collection of nodes and paths of a KG is a category called the *path category* of the KG.

The collection of all knowledge graphs forms a category **KG** whose morphisms are functions that preserve edges of the graph. More precisely, a *morphism* $f: G \to H$ of knowledge graphs G and H is a function $f: G.node \to H.node$ such that

1. For every $x \in G.node$ such that $x \notin Id$, we have that $f(x)=x$,
2. For every $x \in G.node$ such that $x \in Id$, we have that $f(x) \in Id$, and
3. For every $(s, p, o, e) \in G$, we have that $\big(f(s), f(p), f(o), f(e)\big) \in H$.

A *functor* **F** from a category **C** to a category **D**, written **F**: **C** → **D**, is a pair of functions

1. **F.ob**: **C.ob** → **D.ob**
2. **F.hom**: **C.hom** → **D.hom**

such that

a) **C.src;F.ob = F.hom;D.src**,
b) **C.tar;F.ob = F.hom;D.tar**,
c) **C.id;F.hom = F.ob;D.id**, and
d) **C.comp;F.hom = (F.hom ⋈ F.hom);D.comp**.

The axioms for a functor can be diagrammed as in Figure 6.

$$\begin{tikzcd}
\text{C.hom} \arrow[r, "\text{F.hom}"] \arrow[d, "\text{C.src}"'] & \text{D.hom} \arrow[d, "\text{D.src}"] \\
\text{C.ob} \arrow[r, "\text{F.ob}"'] & \text{D.ob}
\end{tikzcd}
\qquad
\begin{tikzcd}
\text{C.hom} \arrow[r, "\text{F.hom}"] \arrow[d, "\text{C.tar}"'] & \text{D.hom} \arrow[d, "\text{D.tar}"] \\
\text{C.ob} \arrow[r, "\text{F.ob}"'] & \text{D.ob}
\end{tikzcd}$$

$$\begin{tikzcd}
\text{C.ob} \arrow[r, "\text{C.id}"] \arrow[d, "\text{F.ob}"'] & \text{C.hom} \arrow[d, "\text{F.hom}"] \\
\text{D.ob} \arrow[r, "\text{D.id}"'] & \text{D.hom}
\end{tikzcd}
\qquad
\begin{tikzcd}
\text{C.hom} \bowtie \text{C.hom} \arrow[r, "\text{C.comp}"] \arrow[d, "\text{F.hom} \bowtie \text{F.hom}"] & \text{C.hom} \arrow[d, "\text{F.hom}"] \\
\text{D.hom} \bowtie \text{D.hom} \arrow[r, "\text{D.comp}"'] & \text{D.hom}
\end{tikzcd}$$

Figure 6: The Functor Axioms

The collection of categories and functors forms a category **Cat**. We are now ready to define an institution (Goguen and R. Burstall, 1983).

An institution 𝒥 consists of the following:

1. A category **Sign** whose objects are called *signatures.*
2. A functor **Sen**: **Sign** ➞ **Set**. For a signature Σ, the elements of **Sen**(Σ).**ob** are called the Σ-sentences.
3. A functor **Mod**: **Sign** ➞ **Cat**^{op}. For a signature Σ, the elements of **Mod**(Σ).**ob** are called the Σ-*models*, and the elements of **Mod**(Σ).**hom** are called the Σ-*morphisms*.
4. A relation $\models_\Sigma$ ⊆ **Mod**(Σ).**ob** × **Sen**(Σ).**ob**, for each Σ ∈ **Sign.ob**, called Σ-*satisfaction*.

The satisfaction relation must satisfy the following axiom:

∀φ:Σ→Ξ ∈ **Sign.hom**, ∀s ∈ **Sen**(Σ).**ob**, ∀ξ ∈ **Mod**(Ξ).**ob**, ξ $\models_\Xi$ **Sen**(φ)(s) iff **Mod**(φ)(ξ) $\models_\Sigma$ s

The KGSQL institution is written 𝒦𝒢𝒮𝒬□, and has the following components:

1. The category **Sign** of 𝒦𝒢𝒮𝒬□ has a single object and morphism. Because **Sign** is a singleton, we omit reference to the unique signature of 𝒦𝒢𝒮𝒬□.
2. A sentence of 𝒦𝒢𝒮𝒬□ is an ask query, and the functor **Sen**: **Sign** ➞ **Set** maps the unique signature to the set of all ask queries.

3. A model of $\mathcal{KGSQ}$□ is a knowledge graph. The functor **Mod**: **Sign** → **Cat**^{op} maps the unique signature to the category **KG**.
4. The satisfaction relation ⊨ is the relation $\{(G,Q)|G$ is a KG, Q is an ask query, and $[\![Q]\!]_G = \text{true}\}$

Now **Sign** in $\mathcal{KGSQ}$□ has only one morphism φ, so it must therefore be an identity morphism. Consequently, in the satisfaction axiom for $\mathcal{KGSQ}$□, **Sen**(φ) is the identity function of the set of all ask queries, and **Mod**(φ) is the identity functor from **KG** to itself. Therefore, the satisfaction axiom for $\mathcal{KGSQ}$□ simplifies to the following:

$$\forall s \in \mathbf{Sen.ob}, \forall \xi \in \mathbf{Mod.ob}, \xi \vDash s \text{ iff } \xi \vDash s$$

which is trivially true. So $\mathcal{KGSQ}$□ is an institution.

Appendix C KGSQL Syntax

```
grammar KGSQL;

root : command;

command : prologue ( selectQuery | askQuery
  | constructQuery | insertRequest | deleteRequest );

prologue : prefixDecl*;

prefixDecl : Prefix NamedGraph Identifier;

selectQuery : Select Variable+ whereClause;

askQuery : Ask whereClause;

constructQuery : Construct patternBlock whereClause;

insertRequest : Insert patternBlock whereClause;

deleteRequest : Delete Variable+ whereClause;

whereClause : Where?
```

```
  '{' patternBlock? ( filter '.'? patternBlock? )* '}';

patternBlock : patternsSameSubject ( '.' patternBlock? )?;

filter : Filter constraint;

patternsSameSubject : ( noun | linkedList ) predicateList?;

predicateList : verb objectList ( ';' ( verb objectList )? )*;

objectList : object ( ',' object )*;

object : noun | linkedList;

noun : resourceOrVariable multiplicity?
  | '[' resourceOrVariable resourceOrVariable? typeUnion ']' multiplicity?;

verb : typeUnion multiplicity?
  | '[' resourceOrVariable typeUnion ']' multiplicity?;

typeUnion : Variable | prefixedName ( '|' prefixedName )*;

resourceOrVariable : prefixedName | typedLiteral | numericLiteral
  | True | False | Variable;

linkedList : '(' ( resourceOrVariable | linkedList )* ')'
  | '[' '(' ( resourceOrVariable | linkedList )* ')' prefixedName ']';

prefixedName : NamedGraph LocalName;

constraint : '(' expression ')' | LocalName '(' expressionList? ')';

expressionList : expression ( ',' expression )*;

expression : conditionalAndExpression ( '||' conditionalAndExpression )*;

conditionalAndExpression : relationalExpression ( '&&' relationalExpres-
sion )*;

relationalExpression : additiveExpression ( '=' additiveExpression
 | '!=' additiveExpression | '<' additiveExpression | '>' additiveExpression
 | '<=' additiveExpression | '>=' additiveExpression )?;

additiveExpression : multiplicativeExpression ('+' multiplicativeExpres-
sion
 | '-' multiplicativeExpression | NatPositive | NatNegative | RealPositive
 | RealNegative )*;

multiplicativeExpression : unaryExpression
 ( '*' unaryExpression | '/' unaryExpression )*;

unaryExpression : primaryExpression | '+' primaryExpression
 | '-' primaryExpression | '!' primaryExpression;

primaryExpression : '(' expression ')' | LocalName '(' expressionList? ')'
 | resourceOrVariable | askQuery;

typedLiteral : Literal Lang? | '[' Literal Lang? prefixedName ']'
 | Literal Lang? '^^' prefixedName;

numericLiteral : Nat | NatPositive | NatNegative | UnsignedReal
 | RealPositive | RealNegative;

multiplicity : '{' integer '}' | '{' integer '..' integer '}';

integer : Nat | NatPositive | NatNegative | '*';

// Lexical Scanner Tokens

// In general, KGSQL is case-sensitive,
// but the following reserved words are case-insensitive:

Prefix : [Pp][Rr][Ee][Ff][Ii][Xx] WS;
Select : [Ss][Ee][Ll][Ee][Cc][Tt] WS;
Ask : [Aa][Ss][Kk] WS;
Construct : [Cc][Oo][Nn][Ss][Tt][Rr][Uu][Cc][Tt] WS;
Insert : [Ii][Nn][Ss][Ee][Rr][Tt] WS;
Delete : [Dd][Ee][Ll][Ee][Tt][Ee] WS;
Where : [Ww][Hh][Ee][Rr][Ee];
Filter : [Ff][Ii][Ll][Tt][Ee][Rr];
```

```
True : [Tt][Rr][Uu][Ee];
False : [Ff][Aa][Ll][Ss][Ee];

// The lexical rules were omitted.
// See kgsql.org/KGSQL.g4 for the full Antlr grammar.
```

Challenges in the Design, Implementation, Operation and Maintenance of Knowledge Graphs

Gary Berg-Cross

RDA/US Advisory Group

Abstract

As a useful information system product, a knowledge graph (KG) must be assembled from many diverse, independently developed sources of information. Sources range from simple text and textual definitions to formal ontologies. Along the path of constructing a KG there are many challenges in the design, assembly and implementation. Data and knowledge challenges, including semantic ones, exist at every step of KG lifecycle processes. These include many recursive steps to align, refine and validate the information product. Internal data management problems such as entity identification and refinement are mixed in with external challenges such as the useful scoping of information. And there are sociotechnical challenges as well, such as the best use of interdisciplinary teams. This article is an attempt to overview some of the issues discussed at the 2020 Ontology Summit and related literature on these challenges.

Introduction

THIS ARTICLE GREW out of some of the knowledge graph (KG) research and development topics discussed as part of the Ontology Summit 2019. Knowledge graphs essentially describe real-world entities (classes and instances) and their interrelations using a graph model. While the phrase "knowledge graph" can be found in the literature going back as far as at least 1972 (Schneider. 1973) and the idea of KGs can be found in the early days of artificial intelligence (AI) systems and semantic nets; the concepts evoked by this phrase have turned into a real activity area of work. One can hope there will be a general acceptance of an effective definition of "knowledge graph," since KGs are now a major focus of applied research to develop convenient, queryable information artifacts made from multiple sources of data and information (Noy *et al.*, 2019). However, there remain many challenges in the KG design, assembly and implementation processes needed to enable a KG to reengineer information from a typically raw, messy, and disconnected state. As a simple totality the original, opportunistic sources of data for a KG, such as on the Web are incomplete, and often hard to query, analyze, and visualize. As they are gathered these must be cleaned and harmonized to a more refined, organized, and linked product that is easier to visualize, query, and analyze. Challenges as part of constructing a KG information artifact exist at every pipeline step of the data lifecycle process. These include many recursive steps within construction to refine and validate the information product. As an information project these are mixed in with external challenges such as scoping of information to use as well as sociotechnical challenges.

In this article I briefly overview these challenges that KGs need to address operationally at significant phases of work, as well as some overall problems that transcend stages. A big picture view of KG development is to see it as similar to system development, running from design, through development and testing to operational deployment. KGs have their own specifics at each stage. Moreover, drilling into phases of work, there are many intersecting steps during development of a knowledge base to support a KG. One may think of some refinement steps as reactions to failed tests of quality that cycle back to some earlier work on knowledge structuring and integration.

For the sake of exposition, phases of work such as entity identification or feature alignment are discussed somewhat separately as part of a typical sequence. Thus, extraction precedes entity identification that leads to entity refinement, and later entity integration and graph completions. Each reflects its own processes and employs distinguishable methods, although they interact and may support one another recursively. It is important to add that due to space limitations, the challenges described in the following sections are not a comprehensive listing of challenges of building and operating a KG system. Rather, they represent issues that were encountered and inspired by discussion topics as part of the 2020 Ontology Summit.

The Mix of External and Internal Challenges along the KG Lifecycle

Developmental steps for a KG start with the external problem of initial data scoping. As part of this domain and data experts identify a core set of the best available data, information and semantic resource sources to be assembled and integrated. This typically reveals a vast space of possibilities to consider and to pare down to a scoped space. Within an established scope, KG development moves from exploring this data space to knowledge and data acquisition. The hope is to acquire both a schema to organize the graph and quality data in order to populate it. But there may be conflicting organizational schemas and/or too little structured data to seed a well-structured graph to start with, and work may proceed bottom up driven by data.

Scoping and Initial Models

Preceding the building of a KG are issues around the scope (and source) of the knowledge needed by the system. To illustrate problems I take here a wide view and put an emphasis on enterprise level KGs (EKG) systems which have larger data acquisition issues than smaller, standalone applications that may have a narrow domain focus.

Scoping analysis identifies the network of potentially relatable entities, such as available from datasets, relational databases, spreadsheets, XML, JSON, Web APIs. Linked Data etc. Together these contain a large amount of structured data together with unstructured raw and poorly documented data that can be leveraged to build and augment a knowledge graph (Blomqvist *et al*., 2010). These should provide a base of knowledge to satisfy competency questions and related queries, which are used as part of guiding methodologies such as eXtreme Design (XD) for building a KG. Because data in the KG's target space is heterogeneous, the scope of EKG coverage typically comes initially without a common model. Moreover, while there are many ontologies now available, usually they do not cover the scope of an EKG. In some cases several overlapping ontologies and/or conceptual models may be available. However, merging and converging these comes with many issues, such as how to manage differing hierarchies or harmonizing definitions and axioms. More details on addressing some of these issues is discussed further in another article in this special edition (Berg-Cross, 2021). It is simple to say in short that developing or adopting a unified model within a projected KG scope is a challenge and is often deferred for a while. Instead, to get underway a loose or lightweight model is often assembled bottom up from various sources or from simple schemas without too deep a consideration of semantic issues. Richer semantic models may be developed along the way or at a later stage of KG maintenance.

Structural issues with Populating and Validating the Knowledge in a KG

Given identification of a scope, efforts proceed to knowledge graph population and curation. Auer *et al.*'s (2018) best practice for population is to use an infrastructure, such as search and extraction tools to access four complementary sources of data/information:

1. First, the infrastructure leverages existing metadata, data, taxonomies, ontologies, and information models. A standard approach for populating a KG, as mentioned before, is to use data stored on the web. Some of that may be unstructured and poorly documented. Not all the data types and relationships will be obvious or correct. Similar concepts even if documented may cover different instances or decompose into different subtypes. Population may fall back on the intuitive semantics latent for human understanding in data labels. As a result, *ad hoc* efforts are often used to explore entity neighborhoods to find candidates for population. These may be as simple as comparing entities and values (Dong, 2020).
2. Second, an infrastructure then provides services, often with graphical assistance that enable direct contributions from scientists who describe their research, supported by intelligent interfaces and automatically generated suggestions.
3. Next it implements some degree of automated methods for information extraction, cleaning and linking.
4. Finally, it supports curation and quality assurance by stakeholders and other interested parties - domain experts, librarians and information scientists.

Whatever level of tooled infrastructure projects have, the acquisition starts with extraction (Dong, 2020). A common wisdom is that the hardest part about building a new KG is everything that happens as data is acquired and before the end product of queries are implemented. However, the fact is that KGs need to extract massive collections of interrelated facts and the underlying data needs to be cleaned. Some automation such as statistical techniques can be used to find anomalies such as misuse of datatype properties in the data (Pujara, Eriq, and Getoor. 2017). Even an initial population of KGs may include many data instances, so they quickly become large enough to hamper the efficiency of the tooled infrastructure for cleaning and checking mentioned above. This also challenges the data quality inspection and testing done by people.

Scoping should have identifying core data, but the next steps dealing with the reality of mapping relationships and understanding key data constraints. Again, a range of automation can help, including natural language processing (NLP) and text or data mining. Extraction from DBs and online linked data is the more familiar part. The structured extraction is similar to querying, while text processing is similar to NLP, but is enhanced by pre-processing. In a pre-processing step, the input, say a collection of online pages with text, can be classified by a template and clustered by another template. For extraction of information from images processing is more like computer vision analysis where one first extracts the numerical features from the text or images and then gives those visual features as input to a machine learning (ML) model. Both types of extraction can provide a candidate base of data extracted facts to work with.

The next phase of work transforms these candidate facts into a large, useful knowledge graph. This is itself formidable due to structural issues. As mentioned, KGs are heavily populated from unstructured data. This is the problem of "noisy data". The state of the art to handle this includes using statistical techniques to enhance data fit, aligning features and entities properly. Besides unstructured sources, others are semi-structured. They are not formal and, on the whole, may not be well-structured. This means, for example, that source extractions, including those using automation, which have some structure like linked data, may reflect overly simplified or inaccurate information. Online semi-structured information such as Wikipedia and the related DBpedia represents such sources that are low hanging targets. Another, crowd-sourced base for building a KG is Wikidata, which has more than 25k active users and employs 329 bots. It is a tempting source since it contains more than a billion statements about 92 million entities (Arnaout, 2021). Parts of these sources, like Wikipedia infobox tables, are often used for early population. However, several major challenges have been noted with this source, including:

1. Deterministic extraction patterns used in DBpedia (and other sources) are dynamic, hence vulnerable to template changes;
2. While links may provide valuable information about a relationship link, the labels may be ad hoc. Too much reliance on labels such as

Wikipedia links can lead to entity disambiguation problems (handling the ambiguity of natural language labels is discussed below);

3. Naive heuristic based extraction of unlinkable entities yields low precision, which hurts both accuracy and completeness of the final KB (Peng et al., 2019).

Both unstructured and semi-structured extractions have to be made to fit together. Moreover, they have to harmonize with the semantic implications each other implies and any formalized knowledge used. Most of the work on automatically mapping structured and semi-structured sources to ontologies focuses on semantic labeling (Qiu *et al*., 2018). But there are challenges. Automated systems such as the Never Ending Language Learner or NELL (Mitchell *et al*., 2018) may extract a fact from Wikipedia with great confidence. However, human and linguistic analysis, such as case analysis, show they can be wrong. Roles an entity plays can be difficult to determine without extensive context. A superficial process sees a role as an "actor" when in reality the entity is a "coach" (Padia, Ankur, 2017). KG developers might make use of linguistic analyses to help overcome the ambiguities of the use of natural language terms labeling data to arrive at formal distinctions for intended interpretations.

Entity/Feature Alignments Entity Refinements, and Naming Resolution

Feature extraction and entity alignment take information from multiple schema types (*e.g*., CSV, data tables) to some form of a common graph-ready schema. This provides organization but may be well short of formal ontology semantics. In the large volume of data spaces, feature alignment and managing entity identities from heterogeneous data sources pose several obstacles even if an underlying model has been crafted. Entity names/labels are often not reliable. Entity resolution, the merging of records that refer to the same entity, is thus a key problem. Do, for example, the labels "born" (say "2001") and "date of birth" (say 5/4/2001) mean the same thing (Pham *et al*., 2016), and do they align with one entity in the model? Linking entity as part of alignment is enabled if a governing schema, hierarchy or conceptual model, such as mentioned above, has been developed. Mapping data to a shared schema or ontology is considered a key step in KG development (Auer *et al*., 2018; Ehrlinger and Wöß. 2016), since aligning with a domain ontology brings the additional benefit of formal semantics, which in turn can help with later alignments.

While building a KG, mature processes are needed to find entity types in unstructured data (Taheriyan, Knoblock *et al.*, 2016). Important information to identify entities and features may also be found in images on the Web. Images are an important and easy source of interpretable, contextual knowledge for humans. However, until recently, automated feature extraction from images was hard. Machine learning techniques using deep neural nets have made a difference, but it is still challenging to align entities and extract feature information from images in a way that is meaningful to humans. It is a sobering fact that feature type ontologies reflecting people's understanding may have 1000 types to choose from (Yan *et al.*, 2021). Moreover, these are often hidden in headings that convey key relations, attributes, or qualification such as the valid dates of facts.

Refinements in ideas such as capturing asserted facts about identity over time from data is a special type of refinement and obviously also challenging. For a KG, alternate sources may claim different, relevant periods and there may be gaps in data that need to be filled wisely. Work with linked data has shown that by analyzing the co-occurrence of topics and entity types, new types can be assigned to entities based on the topics/types found for those entities (Sleeman, Finin, and Anupam, 2015).

Resolving entity identity for moderately sized data sets manually is a bottleneck. By one report, it can take up to 6 months (Hertling and Paulheim, 2018). Some type of entity and name resolution is especially important where alternate textual formulations are used. Again, the size and scale of possible alternative in an information space makes resolutions a challenge. As part of population from DBs, for example, there are too many tables with impossible/incorrect/incompatible labels (who is Doctor "anonymous"?). Moreover, it is hard to join the tables since data was originally modeled for particular applications and not for integration as needed in a KG. Machine learning (ML) automation for names entity resolution (NER) is one place to look for help (Yadavan and Bethard. 2019).

While there remain limits to automated feature and entity resolution, it is also true that extraction using more mentalistic identification and labeling of features can be misleading due to human biases. This is especially likely if extraction is done by data or computer scientists lacking domain experience. Best practice advice by Knoblock (2018) is that it helps to start with sources using semantic labeling that annotates data fields with ontology classes and/or properties. However, a precise mapping that fully recovers the intended entity meaning from data needs to describe the semantic relations between the data fields too. This provides context and shows how good work on one process step to make subsequent steps such as graph integration easier.

There remain entity resolution challenges that come from noisy data of any kind, which raises the question, "Just what is meant?" Knowledge refinement addresses some types of noise like missing knowledge, redundancies or just plain erroneous knowledge. Statistical and ML techniques can and have been used, but we do not yet fully understand the range of possible errors that could occur in extracted facts over various domains (Zou *et al*., 2020). Resolving entity identity remains for now a mixed and balanced process of some automation and some manual, holistic clean up activity.

One should note that there is a big role of ML driven embedding methods in extractions needed to develop KGs. Knowledge graph embedding refers to the embedding of components of a KG including entities and relations into continuous vector spaces. This is used to simplify the refinement of a KG while preserving its inherent structure. Embedding is used by a variety of downstream tasks such as KG completion and relation extraction, and hence has gained some attention as a useful practice. A representative approach embeds KBs into latent spaces and makes inferences by learning and operating on latent representations. Such embedding models, however, do not make use of any rules during inference and hence one suspects have limited accuracy (Lin *et al*., 2015; Wang, Wang, and Guo, 2015; Wang *et al*., 2017).

Graph Integration and Graph Completion

Graph integration relies on handling entity and feature alignment issues, and some early opportunistic integration may happen as part of alignment. If a general schema or ontology has been used integration may take place a bit more routinely. However, early integrations are often only partial and a final phase of integration with validation testing is needed. Without a broad and deep graph integration, we can ask, "Can KG efforts be integrated or are we building silos at a different level?"

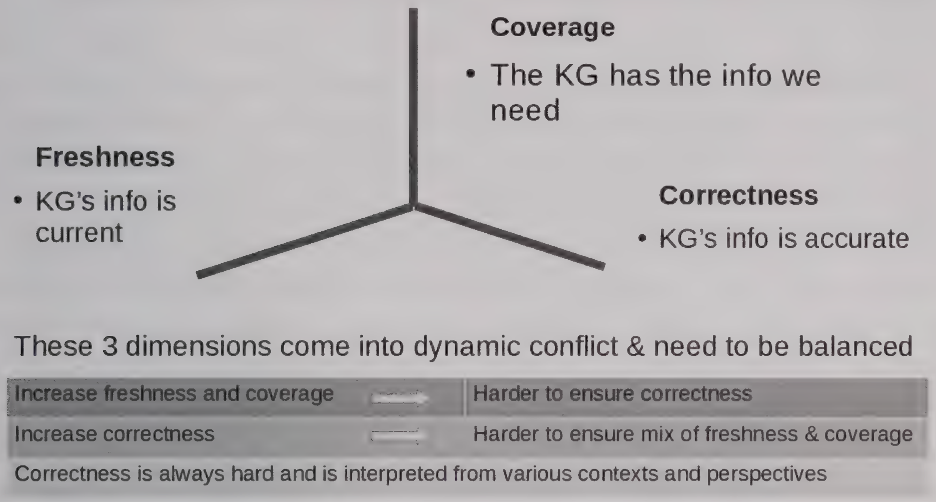
Beyond the immediate problem driven by sources, KG integration difficulties come in several forms. Some are again structural, and some are due to gaps to fill:

- Integration of data coming from a variety of source locations presents challenges beyond the entity naming issues that were previously mentioned. The sources may use different data organizational schemes. Named entities and extracted relations may represent mined information that needs to be further integrated with existing structured data (*e.g.,* via Entity Linking techniques) in order to yield relatively complete entity descriptions;
- One way to view the challenge of graph integration and well as eventual completion is to consider some integration characteristics of the main online sources for population. Wikipedia provides information for a KG early on, but has what is called knowledge incompleteness. For example, in 2014 only 46% of person entities in Wikidata have birthplaces available, according to Vrandeci and Krötzsch (2014). All of this gap filling along with error correction is needed before KG construction is complete.
- The integration be targeted to more than one application, which require different (heterogeneous) data organizational schemes. This is known as KG to App integration. As mentioned, alignment with a community schema or ontology supports this later step of wider integration.

The integration reality for KGs is the need to handle complex relationships as they move from entities to entity integrations. To reflect some of their domain knowledge, KGs can include many complicated relations to handle things as roles and situations that lay things together. These can reflect relations used to capture specific contexts as well as level(s) of abstraction involved in the conceptualizing hierarchies (Kim *et al*., 2015). Importing data for new purposes differing from their original siloed means harmonizing and meaningful mapping entities but also relations between entities. Depending on the degrees of differences in conceptualization, this can make KG integration step a demanding task.

Knowledge graph completion refers to determining if there is a relation between two graph entities and, if so, specifying the type of the relation. One may think of this as completing a triple using 2 entities A and B and finding a relation R to form the triple. Knowledge graph completion may use ML embedding techniques to predict relations between entities under supervision of the existing KG.

One known problem reflecting a KG comprehensiveness, if not completeness, is that many data sources used for entity extraction (per previous examples) can create representativeness problems. As a necessity, large-scale KGs also have to make some type of trade-off between knowledge completeness and correctness. As shown in Figure 1 adapted from Gao (2018) there are really three key KG dimensions in conflict and requiring balance during development. These start with the consideration of correctness and coverage, but also the idea of completeness of a KG along with the timely freshness and veracity of the information. By KG correctness, we don't mean the graph always knows the "right" value for an attribute to answer a query. Rather, it means that the knowledge in a KG is always able or competent to explain why a certain assertion was asserted. The test of correctness is that an assertion should reflect a consensus and make sense to a domain expert. Data provenance about sources captured during data acquisition and documentation of integration trade-offs also provides some explanatory basis for A KG's correctness. Following initial graph population final, tuned construction often includes new links and confidences about facts and relations which advance completeness. This also helps with big data performance, such as the ability of a KG to handle fast data in real-time. However, working, conceptually elegant designs for the first portion of data may not scale up as more instances and types of data are added to complete a KG. Later, the same issues are faced as part of maintenance and updates. As more data is acquired, different vocabularies are introduced and different patterns may encode the same attribute. To mitigate this challenge, as discussed in another article in this special edition (berg-Cross, 2021), graph construction should be viewed as an incremental process with a final assembly that includes a check of semantic relations, if possible, from a guiding ontology.



These 3 dimensions come into dynamic conflict & need to be balanced

Increase freshness and coverage	→	Harder to ensure correctness
Increase correctness	→	Harder to ensure mix of freshness & coverage
Correctness is always hard and is interpreted from various contexts and perspectives		

Figure 1: Three key KG dimensions to balance - adapted from (Gao, 2018)

Knowledge Representation and Query Languages

As previously noted, data populating a KG may be modeled as RDF triples. They are readily available, and the argument for them is that they enabled easy data management and provides a simple way to fold in some semantics for existing data. KGs tend to be less formal than the best ontologies and follow the Linked Data component of the Semantic Web approach, using RDF/RDFS to express simple factual information. However, RDF has obvious limited expressiveness compared to natural language or ontologies. Indeed, as Pat Hayes pointed out, formalisms like RDF lack expressivity. Semantic nets of the 1970s were, almost unilaterally, much more expressive than knowledge graphs or RDF, or any of the other 'graph'-like modern notations. Semantic nets typically had ways of encoding quantifier scopes, disjunction, negation and sometimes such things as modal operators.

Since RDF/RDFS has limited semantic relations, over time KGs based on them encounter the limits to reasoning. This is a problem documented years ago (Sowa, 2011). The long-term implications are clear: the continued use of limited representation require semantic resources like ontologies to boost knowledge expressiveness. A variety of incremental approach to achieve this is discussed in Berg-Cross (2021).

Graph Construction, Performance, Queries, Scalability and Maintenance

Big data comes with the dimensions of volume and velocity. The challenge is to manage changing knowledge due to the fast incremental updates that are feeding large-scale KGs such as KGs that are being scaled up to handle such problems as epidemic and hospital data. Any effective entity-linked KG structure will grow based on its ever-changing expanse of related data. For example, even standard organizational knowledge represented in a KG may merge or split. New scientific discoveries may break an existing concept like coronavirus into subtypes (Paulheim, 2017). Since size makes it impossible to validate and verify KG updates manually, like initial KG population, it is tempting to automate the verification process. Automation maybe especially effective if data structures are not changing, but instance data is being added. Unsupervised and semi-supervised knowledge extraction from unstructured data in relevant domains is one approach. However, these may again be open to many different interpretations of domain knowledge that have to be resolved. Advances needed for some degree of automated support include:

- better domain knowledge representation and reasoning,
- probabilistic models for adjusting graphical structures and
- natural language inferences that can be used to construct an automatic or semi-automatic system for consistency checking and fact verification.

Selection of a graph base and a system for deployment of a KG is the proverbial last mile. It often receives like discussion, but is actually a diverse problem area, including the obvious effect it can have on performance or on associated tools such as graphical interfaces. Real-time KG operations is a major factor in selection. Graph databases are not known for their speed or scalability. In fact, they are generally speaking the smallest and slowest of data model types and rich semantics are not yet easily adapted for dynamic and responsive application and KG environments (Wei, 2018). Scalability is also a KG issue at all phases of the KG life, and notable for performance (but also maintenance). All the graph models noted by (Rajangam, and Annamalai, 2016) share some common limitations. For large knowledge bases, the graphs become too large to perform required operations within convenient time. Moreover, changes in existing knowledge can increase the overhead cost of maintaining the graph's nodes and edges. Scale issues manifests themselves indirectly by affecting other operations, such as managing fast incremental updates to large-scale KGs. Traditionally, to help performance the operative part of the graph stays in the RAM, but multiple threads can access it; and, as with most KG challenges, there are trade-offs to consider. When performing real-time operations, it is necessary to consider the time of execution, but also to respect the quality and precision of execution.

Associated with representation issues, there are graph DB performance concerns with modeling certain data types. Time series, for example, are not well expressed in KGs that use simple RDF. This makes for some *ad hoc* structures with processing issues, but some workarounds and new representations for temporal information have been proposed (Leblay and Chekol, 2018).

Besides sheer query performance, most, if not all KG systems, face the challenge of managing the graphs at scale over time. This requires a proper infrastructure. Obviously, a KG infrastructure must include a scalable graph-storage backend to store information and expose a comprehensive API for interacting with the KG.

There are several graph database languages on the market that address both these query performance and graph maintenance. These include Neo4j's Cypher, Google Cayley, TIBCO, Apache TinkerPop Gremlin, Amazon's Neptune and TigerGraph's GSQL *etc*. Selection involves not only performance, but also which query language and its expressiveness a team is comfortable with. New query languages for KGs (like Cypher) exist, but standardization remains a current area of concern for industry stakeholders.

When considered a graph database, other evaluation factors include: ability to deal with all of the previous problem areas mentioned, such as:

- Schema and modeling flexibility and independence
    - This includes graph management capability to design and execute complex algorithms beyond simple queries to exert efficient and granular control of both the graph query and the graph model elements, *i.e.*, editing of vertex and edge instances as needed.

- Ability to import and leverage complex and semantic sources:
  - ontologies, taxonomies, vocabularies
- Linking such as mapping datasets, vocabularies, *etc*. to the KG structure
- Efficient traversal of graph nodes such using parallel semantic processing (Beneventano and Vincini. 2019)
- Privacy and security
  - Not all the KGs use security mechanisms for access. So it is necessary to classify the KG and use tools for analysis before processing.
- Exploration of data via complex GUIs

External Challenges

Besides these internal, construction, performance and maintenance hurdles, there are external requirements that can frustrate KG success. External challenges identified by Sheth *et al*. (2019) include capturing context and domain-specific knowledge issues. Context exists in KG neighborhood structures, but may not reflect human understanding and reasoning about an entity and its context. The pre-conscious background knowledge and related reasoning engaged in as part of human understanding seems very different from what exists in current KGs. Some context is captured in populating KGs and ranges from spatial and temporal information and related reasoning along with provenance. However, context for human style reasoning, learning and commonsense understanding are largely unaddressed in current work. Instead, the reasoning that is typically available as a part of a KG in simple logical inference, graph node-wise reasoning, such as search, link predication, entity prediction, or subgraph matching (Liu *et al*., 2020). The difficulty of adding commonsense reasoning in particular employed over a large base of commonsense knowledge captured in a KG remains.

Related questions include how to handle implicit relations, strength of (causal) relations, and exclusiveness (Popping, 2003). This is a recognized gap that some hope to start to fill with intelligent text analysis and by the crawling and analyzing of relevant sites and social media in real-time as a better source of commonly understood and used knowledge (Ilievski, Szekely, and Zhang, 2021).

The Internet of Things (IoT), like healthcare, represents an example of an external challenge of interest that Sheth *et al*. (2019) feature. It is a complex domain with many interacting specific subdomain parts that may involve commonsense reasoning and that promise big potential for successful applications. However, an IoT KG comes with specific knowledge issues, including how to automate the building of a large base of domain knowledge and cross disciplinary schemas from text to support a range of applications. A start on this has been made by Noura *et al*. (2019) to test how well existing ontologies in subdomains match up to concepts extracted from IoT text.

To these examples we might add the continuing challenge of adequate visualization as part of user interfaces. Visualization methods remain the main means to support KG usability, analysis if completeness and clarity and to provide both big picture and drill down detail understanding. Early efforts and visualization tools (graph-based or template-based visualization) mainly reflected a self-limited ability to expose the syntactic structure of KGs rather than their conceptual semantics (Desimoni & Po, 2020). They lacked the flexibility to easily visualized diverse parts of a graph or allow users to specify information. Since multiple semantic resources, such as ontologies and ODPs may be used to craft a KG, their visualization is important too. Interfaces that allowed source comparisons of multiple resources were not available until very recently and are still in early stages of development (Asprino, Carriero, and Presutti, 2021). Finally, visualization serves as an explanatory tool. Users often need an interface with a modest explanatory ability to describe the relative important of attributes as part of query relaxation or refinement when displaying results.

Discussion of Opportunities and Future Directions

In this paper, I have reviewed some of the challenges in KG development and use. Despite the challenges, KGs have started to have a significant effect on data and knowledge management in particular areas. This is likely to grow into a more general impact. To achieve this, there remain a large suite of problems with activities that are internal to the KG lifecycle. This includes phases of works from scoping through population, alignment and refinement to completion of graphs. There are potential trade-offs within and between each and every process along the way. Within the KG lifecycle, semantic and data technologies are of obvious value and play a role. A variety of different methods, ranging from NLP and ML techniques, are increasingly used, but they yield approximate products and cannot yet adequately automate solutions to commonsense problems like resolving entity identity over time. There remains a need to address development of common tools that can interoperate across the KG lifecycle. Supervised ML is one area of notable research and is increasingly important in extracting text and images. However, a concern is that there is not enough training data to support robust ML and deep learning systems, especially in complex and promising areas like IoT. Approaches to overcome this can adopt fully unsupervised ML approaches (*e.g.*, clustering with vector representations) or semi-supervised techniques such as distant supervision with existing knowledge, multi-instance learning, active learning, *etc.* (Casolla *et al.*, 2019). While automated help is advancing, real reasoning limitations have been noted. For example, we do not yet know how to integrate logical views with statistical ones. Statistical methods as reflected in deep neural networks do not readily provide information about and for the process of "reasoning" or "deduction". This generates problems for applications, including KGs where explanation and dialog are needed for users and developers. And one can hope that KG developers will understand the need for a well thought out schema and how ontologies, or at least ontological analysis, can aid in KG semantic improvement. All of these challenges within the vision of manageable KG systems and infrastructure architecture handling need to be addressed. It seems likely that there will transitional systems to incorporate KG systems and supporting infrastructure into more traditional and front-line dynamic information systems,

As of yet, scalability issues have not been systematically researched for most aspects of the KG lifecycle, as well as operational performance. Mining and refining associated neighborhoods and paths in large graphs, for example, is only starting to be addressed.

Conclusions

In conclusion while challenges remain, there are now active technical research areas to support the rapidly expanding space of KGs in order to align and semantically unify richly interconnected heterogeneous data. It is encouraging that, despite the difficulty of population efforts using multiple sources, we are becoming better at building KGs with less noise at each phase of work. Among the remaining challenges are those of ontology merging, developing an adequate base for ML, agreeing on an adequate approach to situational and contextual understanding, and understanding how to use deep learning in dynamic situations. It is especially important to support the need to keep humans in the loop with the variety of automation being developed, such as ML-generated models. Moreover, there remains the need for a common, enhanced ontology engineering practice addressing the of structuring the semantics of KGs.

References

Arnaout, Hiba *et al.* "Negative Knowledge for Open-world Wikidata." *Companion Proceedings of the Web Conference 2021*. 2021.

Asprino, Luigi, Valentina Anita Carriero, and Valentina Presutti. "Extraction of common conceptual components from multiple ontologies." *arXiv preprint arXiv:2106.12831* (2021).

Auer, Sören *et al.* "Towards a knowledge graph for science." Proceedings of the 8th International Conference on Web Intelligence, Mining and Semantics. ACM, 2018.

Beneventano, Domenico, and Maurizio Vincini. "Foreword to the Special Issue:"Semantics for Big Data Integration"." (2019): 68.

Berg-Cross, Gary, Issues in Incrementally Adding Better Semantics to KGs. 2021

Casolla, Giampaolo *et al.* "Exploring unsupervised learning techniques for the Internet of Things." *IEEE Transactions on Industrial Informatics* 16.4 (2019): 2621-2628.

Blomqvist, Eva *et al.* "Experimenting with eXtreme Design". In: Proc. of EKAW 2010. (Lisbon, Portugal). Vol. 6317. Springer, 2010, pp. 120–134.

Desimoni, F., Po, L.: Empirical evaluation of linked data visualization tools. Future Generation Computer Systems 112, 258–282 (2020)

Dong, L. "Knowledge graph and machine learning: A natural synergy, presentation at Stanford seminar on KGs." (2020).

Ehrlinger, Lisa, and Wolfram Wöß. "Towards a Definition of Knowledge Graphs." *SEMANTiCS (Posters, Demos, SuCCESS)* 48.1-4 (2016): 2.

Hertling, Sven, and Heiko Paulheim. "Dbkwik: A consolidated knowledge graph from thousands of wikis." 2018 IEEE International Conference on Big Knowledge (ICBK). IEEE, 2018.

Ilievski, Filip, Pedro Szekely, and Bin Zhang. "Cskg: The commonsense knowledge graph." *European Semantic Web Conference.* Springer, Cham, 2021.

Kim, Dokyoon *et al.* "Knowledge boosting: a graph-based integration approach with multi-omics data and genomic knowledge for cancer clinical outcome prediction." *Journal of the American Medical Informatics Association* 22.1 (2015): 109-120.

Knoblock, Craig. From Artwork to Cyber Attacks: Lessons Learned in Building Knowledge Graphs using Semantic Web Technologies, U.S. Semantic Technologies Symposium March 1, 2018.

Leblay, Julien, and Melisachew Wudage Chekol. "Deriving validity time in knowledge graph." *Companion Proceedings of the The Web Conference 2018*. 2018.

Lin, Yankai *et al.* "Learning entity and relation embeddings for knowledge graph completion." *Twenty-ninth AAAI conference on artificial intelligence*. 2015.

Liu L, Du B, Ji H, Tong H. KompaRe: A Knowledge Graph Comparative Reasoning System. arXiv preprint arXiv:2011.03189. 2020 Nov 6.

Mitchell, Tom *et al.* "Never-ending learning." *Communications of the ACM* 61.5 (2018): 103-115.

Noura, Mahda *et al.* "Automatic Knowledge Extraction to Build Semantic Web of Things Applications." *IEEE Internet Things J.* 6.5 (2019): 8447-8454.

Peng, Boya *et al.* "Improving Knowledge Base Construction from Robust Infobox Extraction." *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Industry Papers)*. 2019.

Pham, Minh *et al.* "Semantic labeling: a domain-independent approach." International Semantic Web Conference. Springer, Cham, 2016.

Popping, Roel. "Knowledge graphs and network text analysis." *Social Science Information* 42.1 (2003): 91-106.

Pujara, Jay, Eriq Augustine, and Lise Getoor. "Sparsity and noise: Where knowledge graph embeddings fall short." *Proceedings of the 2017 conference on empirical methods in natural language processing*. 2017.

Qiu, Jing *et al*. "Automatic non-taxonomic relation extraction from big data in smart city." *IEEE Access* 6 (2018): 74854-74864.

Sleeman, Jennifer, Tim Finin, and Anupam Joshi. "Topic modeling for rdf graphs." *3rd International Workshop on Linked Data for Information Extraction, 14th International Semantic Web Conference*. Vol. 1267. 2015.

Edward W. Schneider. 1973. Course Modularization Applied: The Interface System and Its Implications For Sequence Control and Data Analysis. In Association for the Development of Instructional Systems (ADIS), Chicago, Illinois, April 1972.

Gao, Yuqing *et al*. "Building a large-scale, accurate and fresh knowledge graph." *SigKDD*. 2018.

Padia, Ankur. "Cleaning noisy knowledge graphs." *Proceedings of the Doctoral Consortium at the 16th International Semantic Web Conference*. Vol. 1962. 2017.

Sheth, Amit, Swati Padhee, and Amelie Gyrard. "Knowledge graphs and knowledge networks: The story in brief." *IEEE Internet Computing* 23.4 (2019): 67-75.

Sleeman, Jennifer, Tim Finin, and Anupam Joshi. "Topic modeling for rdf graphs." *3rd International Workshop on Linked Data for Information Extraction, 14th International Semantic Web Conference*. Vol. 1267. 2015.

Sowa, John F. "Future directions for semantic systems." *Intelligence-based systems engineering*. Springer, Berlin, Heidelberg, 2011. 23-47.

M. Taheriyan, C. A. Knoblock *et al*., "Leveraging Linked Data to Discover Semantic Relations Within Data Sources," in ISWC, 2016.

Vrandečić, Denny, and Markus Krötzsch. "Wikidata: a free collaborative knowledgebase." *Communications of the ACM* 57.10 (2014): 78-85.

Wang, Quan, Bin Wang, and Li Guo. "Knowledge base completion using embeddings and rules." *Twenty-Fourth International Joint Conference on Artificial Intelligence*. 2015.

Wang, Quan *et al*. "Knowledge graph embedding: A survey of approaches and applications." *IEEE Transactions on Knowledge and Data Engineering* 29.12 (2017): 2724-2743.

Yadav, Vikas, and Steven Bethard. "A survey on recent advances in named entity recognition from deep learning models." *arXiv preprint arXiv:1910.11470* (2019).

Yan, Bo *et al*. "Harnessing Heterogeneous Big Geospatial Data." *Handbook of Big Geospatial Data* (2021): 459.

Zhu, Q., Wei, H., Sisman, B., Zheng, D., Faloutsos, C., Dong, X. & Han, J. (2020) Collective multi-type entity alignment between knowledge graphs. In *WebConf 2020*.

Annual Meeting and Awards Banquet

Outgoing President's Remarks
Mina Izadjoo
President 2021

09-09-2021

Good Evening Everyone,

I would like to thank each one of you for joining us tonight. What a great evening it has been. Beautiful venue, great food, and a beautiful clear sky. The highlight of tonight's gathering is all about recognition of the award recipients for their contribution to science.

Congratulations to all the Awardees! Thank you for your dedication and contribution to the advancement of science.

Science and innovation are for everyone. Science does not see a person's race, religion, or nationality. Science is for anyone who cares about human race and the future of our planet.

The Covid pandemic has made our lives difficult and more challenging but more than ever it highlighted the critical importance of science in the control of this pandemic. It has become obvious that it is the scientists and innovators who can overcome this pandemic.

The Academy's mission since its inception in 1898 has remained the same. The Academy's focus has been to stimulate scientific interest and promote science through collaboration, membership, and publication. In addition to bringing local scientists together, The Academy creates an enabling environment for the youth through its Junior Academy.

It has been a great honor and privilege to have served as the President of the Academy last year. My appreciation and sincere gratitude to the Board Members who unconditionally volunteer their time. Our dedicated Board Members work selflessly and very hard without any financial compensation. This is clearly a testimony to their passion and dedication.

We need your help to continue with our mission of promoting scientific interest not only for this generation but also for the next generation.

We look forward to working with you. Please contact us and learn about how you can work with the Academy.

Incoming President's Remarks
Ram Sriram

Thank you all for attending today's event – both physically and virtually. I am very happy to take over as the president of the academy. We would also like to thank Mina for her excellent job as the previous president and I will try my best to continue her vision for the academy. I will try to keep my comments brief. There are several things that I would like to accomplish this year.

As you all know the Washington Academy of Sciences is one of oldest scientific academies in the United States, going back to 1898. The founders included such luminaries as Alexander Graham Bell and Samuel Langley, secretary of Smithsonian Institution. The primary purpose of our academy is to encourage the advancement of science and to conduct endow, or assist investigation in any department of science." Several Nobel Laureates were members of our academy. We are also fortunate to have Turing Award Winner Vint Cerf as one of our fellow members. Given the past history of our academy, I believe we should strive to keep up the reputation in advancing science. There are several activities we would like to purse the coming year.

Membership. We would like your help in spreading the word about WAS and recruiting new members, particularly the younger generation. We also hope to reach out to other scientific organizations, such as WISE (Women In Science and Engineering). We will also ensure that our fellows are appropriately recognized on our website. Please reach out to Mahesh Mani for your suggestions on how to expand the membership.

Seminars. We hope to have regular virtual seminars from eminent scientists throughout the year. Please visit our website for recorded talks last year. We would also appreciate any suggestions from you on seminar speakers. Please contact President-elect Lynnette Madsen with your suggestions.

Junior academy. Since the 1940's, the Washington Academy of Sciences has sponsored a Junior Academy of Sciences. The Junior Academy exists primarily to serve pre-college schools in DC, Maryland and northern

Virginia. We provide professional scientists who serve as judges in STEM fair (Science, Technology, Engineering, and Mathematics) programs in these schools. The judges attend fair events, interview students and evaluate their projects. We provide Certificate awards based on achievement, as well as feedback to students and teachers for improving science fair quality and guidance. We hope to expand Junior Academy services with more programs, such as opportunities for publication, to reach more students and encourage them to consider careers in STEM. Paul Arveson has been spending considerable amount of time in promoting the Junior Academy. Please feel free to contact Paul if you would like to get involved in this activity.

Journal. WAS also publishes the Journal of the Washington Academy of Sciences quarterly. This journal has been very successful under the stewardship of Sethanne Howard. Authors have included more than ten Nobel prize winners. We hope to receive contributions from you all. Please encourage your colleagues to submit papers to the journal. Information about how to submit a paper can be found at the academy website.

Scientific advisory board. We believe getting the input from a scientific advisory board, whose members are well recognized researchers in the field will guide us toward our goal of advancing science and engineering. We are hoping to put together such a Board this year.

Funding. For several years we have been renting an office from AAAS for a nominal fee. This is going to change soon, as AAAS wants to negotiate a new contract. We hope to retain some office space in the AAAS building. We also believe there is a need for the Academy to obtain additional funds through other contributions. Your help in supporting the activities of the Academy will be greatly appreciated.

My thanks to the Board members for all their help so far. Additionally, Terry Longstreth with his knowledge-base of the academy made sure we followed proper procedures, and Mala Ramaiah took copious notes of the Board meetings throughout the year.

The entire award process was handled by Mahesh Mani and he was very diligent in ensuring that the process was rigorous and fair. Judy Stavely did the work for organizing the Banquet and various other duties associated with the function. Paul Arveson was responsible for setting up the webpage and configuring the equipment for the Zoom conference. All the finances were handled by Ron Hietala, who is the treasurer. Now, a brief introduction to our 2021-2022 Board.

President-Elect: Lynnette Madsen
Secretary: Mala Ramaiah
Treasurer: Ron Hietala
VP Administrative Affairs: Terry Longstreth
VP Affiliated Societies: Parisa Meisami
VP Membership: Mahesh Mani
VP Junior Academy: Paul Arveson
Journal Editor (Ex Officio): Sethanne Howard

Members of Large: Lisa Frehill, Joanne Horn, Parviz Izadjoo, Tanner Ash, Judy Staveley, and David Torain

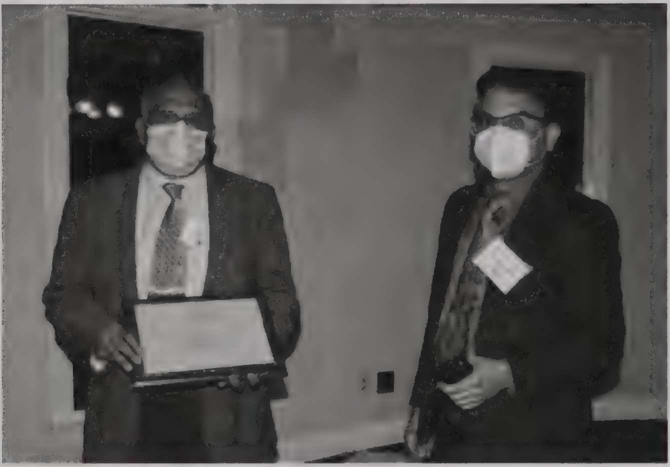
Delegates: Michael Cohen, Jorome (Jerry) Gibbon

Immediate Past President (Ex Officio): Mina Izadjoo

Awardees

Anthony Kearsley
The Award for Excellence in Research in Applied Mathematics

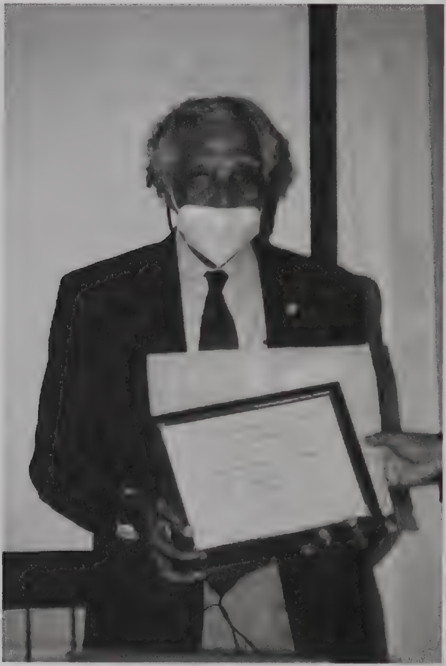
A Y Rao
The Early Career Award in Materials Science and Engineering


Aravind Srinivasan
The Distinguished Career in Computer Science Award

Ayush Varshney
The Early Career Award in Quantum Computing

Breyana Edwards
The Early Career Award in Robotics

Harvey Hack
The Award for Leadership in International Corrosion Standards



James Garrant
Krupsaw Award for Non-Traditional Teaching

Payman Dehghanian
The Early Career Award in Electrical and Computer Engineering

Syed Qadri
The Excellence in Research in Materials Science and Technology Award

Wendy Nilsen
The Leadership in Biological and Computer Science Award

Delegates to the Washington Academy of Sciences Representing Affiliated Scientific Societies

Acoustical Society of America	Paul Arveson
American/International Association of Dental Research	J. Terrell Hoffeld
American Association of Physics Teachers, Chesapeake Section	Frank R. Haig, S. J.
American Astronomical Society	Sethanne Howard
American Fisheries Society	Lee Benaka
American Institute of Aeronautics and Astronautics	David W. Brandt
American Institute of Mining, Metallurgy & Exploration	E. Lee Bray
American Meteorological Society	Vacant
American Nuclear Society	Charles Martin
American Phytopathological Society	Vacant
American Society for Cybernetics	Stuart Umpleby
American Society for Microbiology	Vacant
American Society of Civil Engineers	Vacant
American Society of Mechanical Engineers	Daniel J. Vavrick
American Society of Plant Physiology	Mark Holland
Anthropological Society of Washington	Vacant
ASM International	Toni Marechaux
Association for Women in Science	Jodi Wesemann
Association for Computing Machinery	Vacant
Association for Science, Technology, and Innovation	F. Douglas Witherspoon
Association of Information Technology Professionals	Vacant
Biological Society of Washington	Vacant
Botanical Society of Washington	Chris Puttock
Capital Area Food Protection Association	Keith Lempel
Chemical Society of Washington	Vacant
District of Columbia Institute of Chemists	Vacant
Eastern Sociological Society	Ronald W. Mandersheid
Electrochemical Society	Vacant
Entomological Society of Washington	Vacant
Geological Society of Washington	Jeff Plescia Jurate Landwehr
Historical Society of Washington DC	Vacant
Human Factors and Ergonomics Society	Gerald Krueger

(continued on next page)

Delegates to the Washington Academy of Sciences Representing Affiliated Scientific Societies

(continued from previous page)

Institute of Electrical and Electronics Engineers, Washington Section	Richard Hill
Institute of Food Technologies, Washington DC Section	Taylor Wallace
Institute of Industrial Engineers, National Capital Chapter	Neal F. Schmeidler
International Association for Dental Research, American Section	Christopher Fox
International Society for the Systems Sciences	Vacant
International Society of Automation, Baltimore Washington Section	Richard Sommerfield
Instrument Society of America	Hank Hegner
Marine Technology Society	Jake Sobin
Maryland Native Plant Society	Vacant
Mathematical Association of America, Maryland-District of Columbia-Virginia Section	John Hamman
Medical Society of the District of Columbia	Julian Craig
National Capital Area Skeptics	Vacant
National Capital Astronomers	Jay H. Miller
National Geographic Society	Vacant
Optical Society of America, National Capital Section	Jim Heaney
Pest Science Society of America	Vacant
Philosophical Society of Washington	Michael P. Cohen
Society for Experimental Biology and Medicine	Vacant
Society of American Foresters, National Capital Society	Marilyn Buford
Society of American Military Engineers, Washington DC Post	Vacant
Society of Manufacturing Engineers, Washington DC Chapter	Vacant
Society of Mining, Metallurgy, and Exploration, Inc., Washington DC Section	E. Lee Bray
Soil and Water Conservation Society, National Capital Chapter	Erika Larsen
Technology Transfer Society, Washington Area Chapter	Richard Leshuk
Virginia Native Plant Society, Potomac Chapter	Alan Ford
Washington DC Chapter of the Institute for Operations Research and the Management Sciences (WINFORMS)	Meagan Pitluck-Schmitt
Washington Evolutionary Systems Society	Vacant
Washington History of Science Club	Albert G. Gluckman
Washington Paint Technology Group	Vacant
Washington Society of Engineers	Alvin Reiner
Washington Society for the History of Medicine	Alain Touwaide
Washington Statistical Society	Michael P. Cohen
World Future Society, National Capital Region Chapter	Jim Honig

Washington Academy of Sciences
Room GL117
1200 New York Ave. NW
Washington, DC 20005
Return Postage Guaranteed

NONPROFIT ORG
US POSTAGE PAID
MERRIFIELD VA 22081
PERMIT# 888

5*8**********108*****************AUTO**MIXED ADC 207
HARVARD LAW S LIB ERSMCZ
LANGDELL HALL 152
1545 MASSACHUSETTS AVE
CAMBRIDGE, MA 02138-2903